VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra 456

Univerzální multimediální framework
Universal multimedia framework

Radical Chat

2010 Jaromír Šivic

Zadání

Multimediálně zaměřený framework určený k vývoji bohatých webových aplikací (RIA).

Vytvořte framework pro vývoj webových aplikací sloužících pro publikování multimediálního obsahu na internetu. Systém musí být plně funkční a využitelný na OS Windows i Linux a musí obsahovat následující části:

- Komponenty pro práci se stream serverem - Flash.
- Přehrávač multimediálních souborů.
- Komponenty pro práci v asynchronním módů.
- Video-střižna - software, pro konverzi video souborů.

Výsledná práce musí obsahovat:

1. Analýza problému a návrh řešení
2. Návrh řešení všech dílčích problémů
3. Popis frameworku

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Poděkování

Děkuji vedoucímu mé diplomové práce Panu Ing. Janu Platošovi za podnětné návrhy a celkově vřelou spolupráci při realizaci mé diplomové práce.

Rád bych rovněž poděkoval panu Ing. Pasi Manninenovi z university Jyväskylä University of Applied Sciences za odborné rady a pomoc při testování finální verze multimediálního frameworku jemuž se tato práce věnuje.

…… ……

V ostravě dne Jaromír Šivic

Abstrakt

Tato práce se zabývá analýzou, návrhem a implementací univerzálního multimediálního frameworku určeného pro vývojáře online videokonferencí a aplikací pro prezentaci živého multimediálního obsahu.

Výsledkem této práce by měl být široce použitelný systém pro vytváření, administraci a správu živé audiovizuální komunikace založené na protokolu RTMP, nebo RTMFP. Výsledné řešení umožňuje PHP vývojářům a vývojářům na platformě .NET (C# ASP.NET) vyvíjet online webové videokonference, telemosty v rekordním čase, bez nutnosti vědět cokoliv o technologii Adobe Flash, programovacím jazyku ActionScript a technikách programování paralelních, distribuovaných aplikací. Tento multimediální framework je navržen univerzálně, aby jej bylo možné použít i v jiných programovacích jazycích. Integraci a kooperaci všech komponent ve vysoce heterogenním prostředí zajišťují webové služby spolu s Real Time Messaging Protokolem pojaté netradičním způsobem.

Tato práce rozšiřuje již existující projekt Radical Chat, který vytvořil v letech 2008, 2009 Jaromír Šivic na univerzitě Jyväskylä University of Applied Sciences.

Klíčová slova

Adobe Flash Player, ActionScript 3, Java, Red5, Wowza media server, RTMP, PHP, Flash Chat.

Abstract

This thesis describes the implementation of Universal multimedia framework for online videoconferencing.

The final result should present the first real suitable solution for managing and mastering live audio-video communication over RTMP protocol (used by Adobe Flash) from such programming languages as PHP or ASP.NET (C#). Multimedia videoconferencing framework is designed and implemented by universal way to enable almost any platform and programming language to use it. Officially supported technologies are PHP and ASP.NET. Integration and cooperation of all components together in this highly heterogeneous environment are supported by Web Services technology, SOAP protocol and Real Time Messaging Protocol.

This solution allows (PHP,ASP.NET) developers to create live Flash multimedia applications without any knowledge about Flash, ActionScript 3.0, RTMP protocol or media server API written in Java. It extremely speeds up the process of development and deployment of final applications with an increased emphasis on performance of the system. Two main programming languages (ActionScript 3, Java) are used for framework implementation and another two (PHP and ASP.Net – C#) for demonstration, how to write applications in this universal multimedia videoconferencing framework. The client or user of this framework may be anyone who has an internet-capable device with installed Adobe Flash Player 10 compatible plug-in.

Key words

Adobe Flash Player, ActionScript 3, Java, Red5, Wowza media server, RTMP, PHP, Flash Chat.

Seznam použitých symbolů a zkratek

AS3(ActionScript) – Skriptovací jazyk používaný platformou Adobe Flash.

ASP – Action server pages

AJAX – Asynchronous JavaScript and XML

CPU – Central Processing Unit

FLV – Flash Video

FPS – Frames Per Second

FMS – Flash Media Server

GPL – General Public License

GUI – Graphics User Interface

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

IIS – Internet Information Services

LDAP – Lightweight Directory Access Protocol

PHP – Personal Home Pages

RAM – random-access memory

RTMP – Real Time Messaging Protocol

WPF – Windows Presentation Foundation

IDE – Integrated Development Environment

RCSL – Radical Chat Skinning Language

Obsah

Seznam ilustrací

Úvod

Nástin problému

V moderní době hrají Internet a mobilní sítě hlavní roli při spojování, komunikaci lidí a přenášení různorodých informací mezi nimi. I když by se mohlo zdát, že elektronická pošta, textové chaty, sociální sítě a online služby umožňující sdílení videí a jiného multimediálního obsahu dominují dnešnímu modernímu světu – existují specifické důvody, proč není vhodné používat tyto technologie v komerční podnikové sféře.

Drtivá většina emailové korespondence je stále (velice nerozvážně) přenášena sítí v nezašifrované podobě a může být velice jednoduše odposlouchávána. Totéž platí i pro sociální sítě jako je Twitter, či Facebook a jiné dnes již zastaralé textové chaty. Navíc výše zmíněné komunikační prostředky neposkytují komfort a rychlost odezvy jako je tomu v případě audio hovoru nebo konverzace tváří v tvář. V oblasti všeobecně dostupné živé video komunikace, jsou (na počátku roku 2010) nejvíce používány technologie Skype, Google talk a mobilní 3G video-hovory, z oblasti sdílení tzv. on-demand videa je to portál YouTube.

Odborníci se shodují, že veškeré výše zmíněné technologie sice splňují očekávání běžných uživatelů, nicméně bylo by velice neuvážlivé spoléhat se na ně z pohledu komerční sféry – firem a nadnárodních korporací. Zásadním problémem je zde fakt, že všechny výše zmíněné technologie jsou vytvářeny, spravovány a provozovány třetími stranami (cizími společnostmi), u kterých mnohdy není možné dohledat, jak s citlivými daty zacházejí. Dalším podstatným důvodem, proč tyto technologie nepoužívat je jejich nepřizpůsobivost a nemožnost 100% integrace do firemních informačních systémů, či sjednocení s konkrétní firemní politikou. To v praxi znamená, že každá větší společnost, která chce přes internet provozovat meetingy, či publikovat jiný multimediální obsah musí vyvinout značné úsilí a vytvořit svůj vlastní systém, který toto zabezpečí.

Návrh řešení

Tato práce se zabývá návrhem a implementací “Univerzálního multimediálního frameworku”, pomocí něhož je možné vytvářet online videokonference, meetingy, telemosty a aplikace pro publikování živého audia a videa. Framework nese pracovní označení Radical Chat. Jedná se o komplexní nástroj, který umožní vývojářům, obecně na jakékoliv platformě a v téměř libovolném programovacím jazyce vyvíjet videokonference a jiné bohaté internetové aplikace v naprosto rekordním čase.

Pro realizaci navrhovaného řešení je použito nejmodernějších prostředků a technik (které jsou dostupné počátkem roku 2010). Samotný systém Radical Chat přichází s několika novými programovacími technikami a myšlenkami, které dosud nebyly nikde jinde použity. Při realizaci je kladen důraz na efektivitu, stabilitu, jednoduchost systému z pohledu koncových vývojářů, kteří jej budou používat. A také na cenu systému a čas, který bude nutné vynaložit při jeho nasazení.

Postup práce

Tato práce se snaží čtenáři přiblížit prostředí univerzálního multimediálního frameworku zaměřeného na tvorbu videokonferencí a práci s videem. Pokud je to možné, pak je v textu použita pravděpodobně nejprogresivnější vysvětlující a učební metoda - “learn by exmple”. Jedním z předpokladů pro pochopení této práce je základní znalost problematiky webových služeb a principů tvorby bohatých internetových aplikací.

První kapitola se zabývá definicemi obecných výrazů a pojmů. Druhá kapitola pak výběrem platformy a nástrojů v nichž bude univerzální multimediální framework Radical Chat realizován. Podrobně rozebírá výhody a nevýhody všech existujících možností. třetí kapitola se zabývá analýzou vnitřních funkcí a vnějšího rozhraní frameworku. Ve čtvrté kapitole je důkladně popsán návrh a implementace grafických prvků. Vše je doplněno o příklady použití jednotlivých funkcí. Na čtvrtou kapitolu navazuje kapitola pátá pojednávající o dynamickém skinování. Šestá kapitola se zabývá výkonem a testováním jednotlivých funkcí. Kapitola sedmá shrnuje dosažené cíle a možnou budoucnost projektu.

1 Multimediální Framework

1.1 Co je to multimediální framework

Multimediální framework (MMF) nebo též rozšířený multimediální framework zaměřený na videokonference (MMVCF – multimedia videoconferencing framework) – je ucelené softwarové dílo (soubor knihoven a funkcí), které výrazným způsobem urychluje a zjednodušuje vývoj multimediálních aplikací, jejich správu a nasazení do provozu. Dobrý multimediální videokonferenční framework poskytuje vývojářům intuitivní API, zapouzdřuje složité funkce a komunikaci jednotlivých vnitřních komponent. „Dobrý multimediální framework poskytuje ucelené rozhraní pro přístup k různým audio-video formátům a možnost sdílení živého multimediálního obsahu.“ [1]

Multimediální frameworky bývají obvykle používány aplikacemi, jako jsou: přehrávače videí (video players), videokonferenční systémy (videoconference systems), kamerové a bezpečnostní systémy, moderními instant messengery a dalšími.

Má-li být MMVCF universální, pak by mělo být možné použít jej na libovolném operačním systému a jeho API by mělo být zpřístupněno co nejširšímu počtu vývojářů.

1.2 Nejpoužívanější multimediální frameworky současnosti

Mezi nejpoužívanější multimediální frameworky současnosti patří DirectX, FFMpeg, VLC, Java Media Framework a Adobe Flash.

1.2.1 DirectX & Silverlight

DirectX, konkrétně jeho část DirectShow, DirectPlay, DirectMedia nabízí vývojářům sadu funkcí umožňujících zaznamenávání, přehrávání a streamování živého nebo již dříve natočeného videa. Jedná se o rozsáhlý framework vyvíjený společností Microsoft (aktuální verze DirectX 11) a používaný především v počítačových hrách, ale také programy pro práci s videem. Je určen k použití výhradně s operačním systémem Windows. Škála podporovaných audio-video formátů je ve výchozím nastavení relativně malá a omezuje se pouze na zastaralé kodeky s nízkou nebo malou úrovní komprese a na proprietární formát wmv (Windows Media Video). DirectX je nejčastěji používán programátory z prostředí .NET nebo C++.

1.2.2 FFMpeg

Na rozdíl od velice komplexního DirectX se framework FFMpeg zaměřuje pouze na převádění videí mezi různými formáty a základní úpravy obrazu. Jeho velikou předností je nezávislost na platformě (operačním systému, či hardwaru) a rozsáhlá množina všech podporovaných kodeků. FFMpeg je distribuován ve formě zdrojových kódů – jedná se o otevřený(open source) freewarový projekt.

1.2.3 VLC

VLC je podobně jako FFMpeg nezávislý na platformě. Jádro projektu VLC do jisté míry čerpá ze sesterského FFMpeg. VLC je spíše určen pro přehrávání a streamování nežli k převodu mezi formáty a úpravou videa.

1.2.4 Java Media Framework (JMF)

Je knihovna určená především programátorům v jazyce Java a vývojářům Java Appletů. JMF přináší řadu zajímavých funkcí pro přehrávání a zaznamenávání audia a videa, nicméně nepodporuje moderní kodeky. Jeho nezávislost na používaném operačním systému je mnohdy diskutabilní.

1.2.5 Adobe Flash (Flex)

Je unikátní na platformě nezávislý framework vyvíjený společností Adobe, dříve Macromedia, který sice nevyniká širokou škálou podporovaných audio-video formátů, ale jednoznačně patří mezi celosvětově nejpoužívanější multimediální framework. Plugin Adobe Flash Player je aktuálně nejrozšířenějším a nejčastěji instalovaným pluginem do webových prohlížečů. Framework se zaměřuje na pokročilou práci s 2D grafikou a obsahuje také základní funkce v oblasti práce s 3D grafikou. Aplikace pro technologii Flash je nutné vyvíjet v proprietárním programovacím jazyce ActionScript. Poslední verze Frameworku Flash a Flex podporuje přehrávání zvukových souborů MP3, AAC, Nelly moser, Speex a video souborů typu H.263, On2 Vp6 a H.264 (brzy také VP8). Navzdory ne příliš bohaté množině podporovaných audio-video formátů nabízí Flash vynikající možnosti streamovaní a přehrávání živého multimediálního obsahu – k tomu využívá protkolů RTMP(jehož specifikace byla zveřejněna v létě roku 2009) a protokolu RTMFP(jehož specifikace je prozatím tajná).

1.3 Universální multimediální videokonferenční framework

Na rozdíl od frameworků zmíněných v Bodě 1.2 musí universální multimediální framework se zaměření na videokonference splňovat mnoho dalších kriterií. Především by měl být na platformě (operačním systému) nezávislý. Mělo by jej být možné spustit na mobilních telefonech a chytrých kapesních zařízeních. Dalším logickým požadavkem je, aby jeho služeb mohlo využít co nejširší

spektrum vývojářů. Měl by nabízet jednoduché až intuitivní rozhraní pro snadnou tvorbu aplikací. V tomto případě videokonferencí a multimediálních přehrávačů.

V současné době existuje jen pár aplikací které mají základní rysy MMVCF. Jsou psány buď v jazyce C++, nebo postaveny na technologii Adobe Flash, či Java Appletech. Mezi nejznámější patří "Big Blue Button" freewarový produkt, určený především pro elearning, "Adobe connect proffesional" nebo například "123 Flash Chat" – jedna z nejrozšířenějších komerčních aplikací pro online video chatting. Bylo by velice zrádné považovat takovéto programy za univerzální multimediální frameworky zaměřené na videokonferencing, neboť nesplňují některé důležité předpoklady. Především neumožňují vývojářům plně ovládnout vzhled a chování aplikace. Tyto programy jsou většinou vázány na jisté konkrétní běhové prostředí a programovací jazyk. Jsou propojeny s databázemi a logovacími systémy, které mají přesně definovanou strukturu. Samotné aplikace jsou velice robustní, obsahují velké množství rozšířitelných (dokoupitelných) modulů, které ne vždy naplní očekávání vývojářů. Co je ale hlavní - tyto aplikace jsou již z 99% předprogramovány a neposkytují vývojářům dostatečný prostor pro seberealizaci a efektivní dynamickou interakci se systémem – staví je do role běžných administrátorů, kteří mohou systém konfigurovat, ale neovládají jej.

1.4 Specifikace požadavků kladených na universální MMVCF

Jak je z názvu a zadání této práce patrné, zabývá se analýzou, návrhem a především implementací pravděpodobně prvního universálního multimediálního frameworku zaměřeného na videokonferencing a problémy s ním spojené. V bodě 1.3 byl obecně definován pojem univerzální MMVCF. Nyní nadešel čas určit priority a požadavky na výsledek, který by měl vzejít z této práce. Očekávaným výsledkem by měla být všeobecně použitelná knihovna funkcí, která umožní vývojářům (obecně libovolného programovacího jazyka) naprogramovat online videochat, nebo multimediální přehrávač v řádu hodin. Největší důraz je kladen na použitelnost a stabilitu řešení. Výsledné aplikace musí být dostupné co největší skupině uživatelů. Aplikační programové rozhraní musí být dostatečně jednoduché, aby s ním zvládli pracovat i programátoři začátečníci a maximálně dostupné všem různým skupinám vývojářů. Výsledný produkt by měl být do budoucna rozšířitelný.

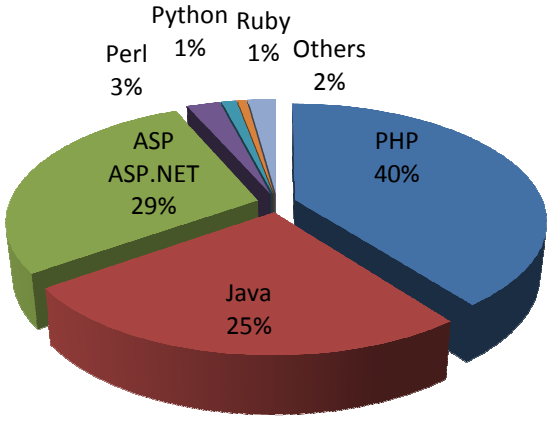
2 Výběr technologie vhodné pro realizaci frameworku

Tato kapitola se zabývá výběrem technologií a platformy v níž bude celý universální multimediální videokonferenční framework (Radical Chat) realizován. Pokouší se identifikovat vývojáře a uživatele koncových aplikací.

2.1 Identifikace uživatelů a vývojářů

K tomu, aby byl budovaný MMVCF skutečně univerzální a mohl se efektivně prosadit bude nezbytné věnovat zvýšenou pozornost aktuální situaci na poli multimediálních aplikací. Je nutné vzít na zřetel soudobé požadavky vývojářů, ale také uživatelů koncových produktů. Jelikož se v moderním světě prosazují především online webové aplikace na úkor desktopových, bude se i framework Radical Chat snažit přiblížit své API co nejširší skupině uživatelů a vývojářů tohoto odvětví. To samozřejmě neznamená, že by nebylo možné použít Radical Chat k vývoji ryze desktopových aplikací.

Zamýšlenou skupinou, kterou chce framework oslovit jsou především vývojáři webových aplikací, informačních systémů a online webových služeb. Na poli těchto odvětví patří k nejpoužívanějším programovacím jazykům PHP, ASP.NET a JAVA.



Obrázek 1 – technologie používané na webových portálech

Pochopitelně mezi podporované platformy (operační systémy) musí patřit Microsoft Windows (alespoň větev NT počínaje operačním systémem Windows 2000), dále pak Linux (jádro 2.6 a

vyšší), Unix BSD a Mac OS. Podpora chytrých telefonů a mobilních kapesních zařízení by měla být do systému Radical Chat také zahrnuta.

2.2 Přehled dostupných technologií

Pravděpodobně nejdůležitější otázkou, která musí být zodpovězena, ještě předtím než bude zahájen návrh frameworku, je: jakou konkrétní technologii použít k implementaci jádra MMVCF a navazujících subsystémů. S tím se pojí i podotázka: která ze současných technologií má největší perspektivu do budoucna a může splnit největší počet předpokladů stanovených v bodě 1.4.
V úvahu přicházejí následující tři možné alternativy.

1. Použít platformu Java a Java Appletů.
2. Systém může být postaven na technologii Microsoft .NET a Windows Presentation Foundation for rich internet applications známější pod názvem Silverlight.
3. Další skvělou alternativou by bylo využít možností Adobe Flash Playeru ve spojení s Flash Media Interactive Serverem.

Každou z výše uvedených technologií je teoreticky možné použít k implementaci dobrého MMVCF. Výhodami a nevýhodami jednotlivých možností se zabývá několik následujících odstavců.

2.3 Platforma Java

Použití platformy Java a Java appletů pro realizaci MMVCF by zcela jistě bylo dostačujícím řešením. Podpora Javy a Java Appletů je v moderním světě webových prohlížečů relativně velká stejně jako rozšířenost JSP stránek na straně serverů. Je pravdou, že dříve měla Java problémy s výkonností obzvláště, co se týče grafického uživatelského prostředí na operačním systému Windows. Dnes jsou naštěstí zmiňované problémy minulostí. Přesto je nutné zastavit se nad několika spornými body, které tuto platformu nepředurčují jako příliš dobrého kandidáta v němž by měl být implementován MMVCF.

Především - Java v posledních letech zaznamenala ústup v počtu instalací na straně desktopových počítačů. Stále ji sice nalezneme na 65% domácích osobních I firemních PC a notebooků – nicméně zmiňované číslo by mohlo být podstatně větší. Dalším velikým problémem konkrétně – Java Appletů je (ve většině případů) nemožnost použít GPU (grafický procesor) pro akceleraci přehrávání multimediálního obsahu. Také je nutné zmínit, že (pro Java Applety) neexistuje žádný dostatečně dobrý výchozí proprietární protokol, nebo transportní vrstva, která by zabezpečila přenos živého audia a videa z jednoho počítače na počítače dalších uživatelů. Navíc datový tok by

byl relativně velký a latence (zpoždění signálu od okamžiku zaznamenání z kamery, přenosu po sítí do okamžiku zobrazení) značná.

2.4 Microsoft .NET & Silverlight

Zajímavé vlastnosti nabízí platforma .NET a technologie Windows Presentation Foundation for rich internet applications známá jako Silverlight. Technologie Silverlight zaznamenala v posledních letech značný rozkvět a velmi se rozšířila mezi všemi uživateli internetu. Mezi její hlavní výhody patří podpora nejnovějších multimediálních formátů jako WMV3, podpora práce s 3D grafikou, hardwarová akcelerace a spolupráce s nejpoužívanějším stream server WMS (Windows Media Server).

Hlavní nevýhodou technologie Silverlight je její silná závislost na platformě. Silverlight a .NET framework sice funguje dobře na operačních systémech Microsoft Windows a v Internetovém prohlížeči Internet Explorer, ale v prostředí Linux, nebo Mac OS jeho podpora značně pokulhává. Aktuální verze Silverlight 3.0 navíc nepodporuje záznam a streamování multimediálního audio-video obsahu z počítačů koncových uživatelů, což je jeden z hlavních úkolů, který má framework Radical Chat řešit. Silverlight, tak může fungovat pouze v roli přehrávače multimediálního obsahu streamovaného z Windows Media Serveru nebo jiným stream serverem kompatibilním s WMS (Darwin nebo Wowza media server).

2.5 Adobe Flash

Technologie Adobe Flash je jednou z posledních možností nabízejících většinu vlastností požadovaných pro tvorbu universálního MMVCF. K hlavním přednostem Adobe Flash a Flash playeru patří široká škála podporovaných zařízení a webových prohlížečů – od desktopů, přes tablety až po mobilní telefony a chytrá kapesní zařízení, na kterých běží Android nebo Windows Phone.

Plugin do webových prohlížečů Adobe Flash Player je v současnosti přítomen na 98% všech počítačů majících permanentní přistup k internetu. (Viz. http://www.adobe.com/products/player_census/flashplayer/). Umožňuje přehrávat videa moderních formátů, ale také je zaznamenávat. Technologie Flash je vybavena vlastním netypovým skriptovacím jazykem ActionScript. Nejnovější verze ActionScript 3.0 je již plně objektově orientovaná. Kód je kompilován do bytekódu a vykonáván Just In Time kompilátorem. To přináší až 50 násobné zrychlení oproti předchozí verzi jazyka ActionScript 2.0. Flash Player (verze 9.0.124 a vyšší) umí využít grafický procesor pro přehrávání multimediálního obsahu a zobrazování 2D, 3D

grafiky. Obsahuje podporu proprietárního protokolu RTMP, který slouží k synchronizaci dat mezi počítači jednotlivých klientů, ale také ke sdílení živého obrazu snímaného z webových kamer. Výhod protokolu RTMP je bohužel (lépe řečeno bylo) možné využít pouze ve spolupráci s Flash Media Interactive Serverem. Popis protokolu RTMP byl dlouhou dobu udržován v tajnosti až do léta roku 2009, kdy společnost Adobe uvolnila jeho specifikaci. Jednou z posledních novinek (kterou přinese teprve Flash Player 10.1) je podpora nového protokolu RTMFP. RTMFP na rozdíl od RTMP umožňuje přímé propojení dvou počítačů a především umožňuje multicastování živého videa – založené na podobném principu jakým funguje protokol BitTorrent.

Hlavní nevýhodou technologie Adobe Flash – hovoříme-li o tvorbě videokonferencí – je nutnost vlastnit, nebo si na některém z existujících hostingů pronajímat služeb Flash Media Interactive Serveru. Průměrná cena Flash Media Interactive Serveru se koncem roku 2009 pohybuje na hranici $5000 za jednu instalaci.

2.6 Zhodnocení dostupných technologií

Rozdíly mezi jednotlivými platformami uvedenými výše jsou zřejmé, přesto však není vůbec jednoduché mezi nimi vybrat tu správnou pro implementaci jádra univerzálního MMVCF. Každá varianta má své klady, ale i zápory. Zvolit k realizaci technologii Java Appletů se může jevit v dnešní době poněkud zpátečnické. Plugin Silverlight nabízí téměř všechny vybrané vlastnosti nezbytné pro realizaci multimediálního frameworku. Chybí zde bohužel klíčová podpora snímání živého obrazu z webové kamery, či TV tuneru (na stanicích koncových uživatelů). Dá se ale (bez nadsázky) očekávat, že Microsoft i tento nedostatek během několika následujících let odstraní. Jediným problémem, který zůstane - bude malá rozšířenost a špatná, nebo nulová podpora Silverlightu na jiných operačních systémech (Linux, Mac OS, Symbian, …). Plugin Adobe Flash Player by se jako hlavní komponenta celého universálního MMVCF frameworku vyjímal ještě mnohem lépe než Microsoft Silverlight. Technologie Flash již dnes (počátek roku 2010) splňuje všechny základní vlastnosti nezbytné pro úspěšný vývoj univerzálního MMVCF. Velikým problémem je ale cena, kterou by každý budoucí uživatel musel vynaložit na zakoupení jedné licence Flash Media Interactive Serveru (Viz. odstavec 2.5). Před zahájením vývoje multimediálního systému je nutné počítat i s tím, že jej v budoucnu budou chtít využít i menší firmy, nebo soukromníci, kteří zkrátka nemohou vynaložit téměř $5000 za FMIS na zahájení provozu multimediálního frameworku sloužícího pro přehrávání videí a video-chatování. Tato částka je neúměrně vysoká. Naštěstí v současné době existují stream servery, které je možno pořídit za výrazně nižší cenu, nebo dokonce zcela zdarma. Vhodnou alternativou k FMIS je například Wowza Media Server nebo Red5.

Po pečlivém uvážení všech kladů a záporů jednotlivých technologií bylo rozhodnuto, že multimediální videokonferenční framework Radical Chat bude jako klíčovou komponentu používat Flash Player a na pozici stream serveru bude využito služeb Wowza Media Serveru a Red5 stream serveru. Jedná se o vskutku neobvyklou kombinaci, která však má své logické opodstatnění.

2.7 Wowza Media Server

Jedná se o komerční alternativu k Flash Media Interactive Serveru vyvíjenou společností Wowza Media Systems. Cena Wowza Media Serveru (WMS) je ve srovnání s FMIS pětinová přitom zůstává zachováno téměř kompletní "media server API" (aktuální verze WMS 1.7 update 12). Server od společnosti Wowza je kompletně implementován v jazyku Java, rovněž serverové aplikace se píší také v jazyku Java, nikoliv ActionScript. Toto řešení je značně nestandardní, ale má své výhody. „Velkou předností je kompatibilita se všemi různými druhy operačních systémů na kterých může běžet. Další výhodou je podpora vícejaderných 64 bitových procesorů. Velice zajímavým modulem je "H.264 non-Flash RTSP/RTP streaming" – umožňující publikování zaznamenávání a následnou úpravu živého videa z IP kamer nebo programů jako je například VLC player. Více o srovnání WMS s FMIS je možné zjistit na webové stránce http://www.wowzamedia.com/comparison.html." [2]

2.8 Red5

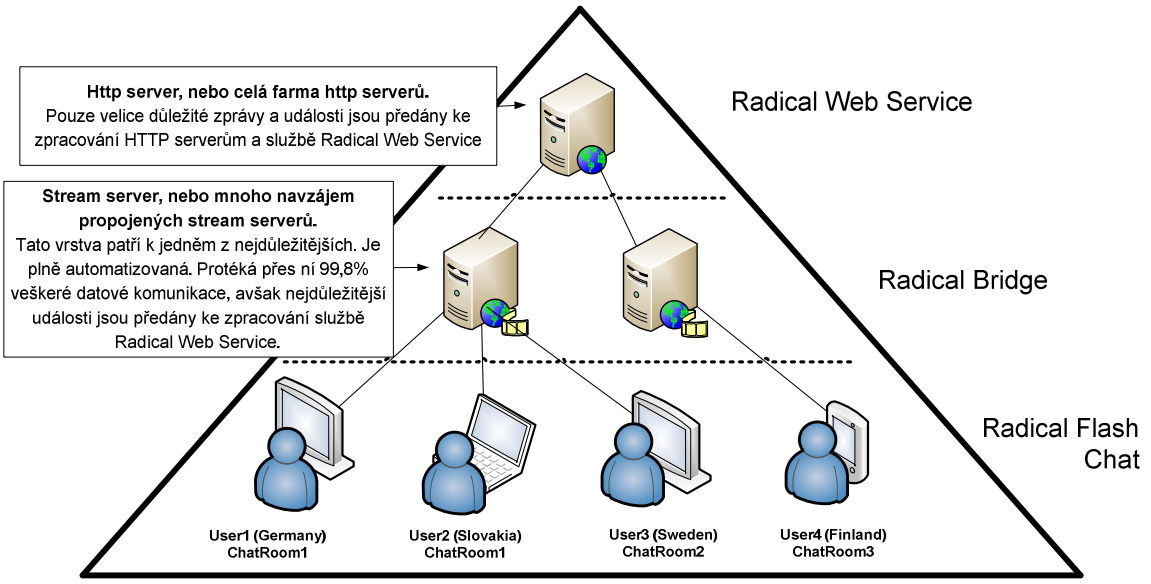
„Red5 je podobně jako Wowza Media Server interaktivním streamovacím serverem určeným pro synchronizaci a sdílení živého multimediálního obsahu mezi Flash Playery v síti Internet. Stejně jako WMS i Red5 je psán v Javě. Nespornou výhodou oproti WMS je ale cena. Red5 je vyvíjen pod licencí GNU-GPL. Jedná se o volně šiřitelný software, jeho zdrojové kódy jsou ke stažení zdarma."[3] Na druhou stranu Red5 je vývojově přibližně dva roky pozadu oproti Wowza Media Serveru. Aktuální verze 0.8.0 nepodporuje streamování ve formátu H.264 ani non-Flash RTSP-RTP streaming. Proto bude univerzální multimediální framework, který vzejde z této práce, nejprve portován na Wowza Media Server, teprve až posléze na Red5.

3 Návrh univerzálního multimediálního frameworku

Tato kapitola se zabývá návrhem universálního multimediálního frameworku. Analyzuje jej a provádí detailní rozbor jednotlivých částí.

3.1 Třívrstvá architektura

Celý framework se z pohledu vývojářů a koncových uživatelů může jevit jako jednolitý celek. Ve skutečnosti se však jedná o velice komplexní, heterogenní dílo, kde jednotlivé softwarové komponenty mohou být umístěny na různých (hardwarových) serverech. Aby toho nebylo málo jednotlivé části jádra frameworku Radical Chat jsou implementovány v různých programovacích jazycích (Konkrétně ActionScript, Java, PHP a ASP.NET), což přináší netušené možnosti použití, ale značně to komplikuje návrh a implementaci jádra MMVCF. Po pečlivém uvážení byla zvolena koncepce třívrstvé architektury.



Obrázek 2 - třívrstvá architektura [4]

Celý framework nese název Radical Chat a dělí se do tří hlavních částí – “Radical Flash Chat”, “Radical Bridge” a “Radical Web Service”. Nejnižší vrstvu (Radical Flash Chat) tvoří relativně malá aplikace, psaná v jazyku ActionScript 3.0 určená pro Adobe Flash Player 9.0.0 a vyšší. Jedná se o

jedinou část celého systému, která je přístupná běžným uživatelům. Většinou běží ve webovém prohlížeči jako zásuvný modul webové stránky, nicméně může být spuštěna i jako samostatná desktopová aplikace.

Radical Flash Chat (RFC) je propojen se stream serverem a aplikací Radical Bridge (RB). Ten řídí chování RFC, filtruje příchozí a odchozí zprávy a reaguje na chování uživatelů. Radical Bridge je psán v jazyku Java a je propojen se stream serverem Wowza Media Server se kterým tvoří jednolitý celek. Později bude přidána také podpora Red5. Jelikož se jedná bezpochyby o nejzatěžovanější část celého systému – je kladen veliký důraz na stabilitu a efektivitu celého stream serveru. Radical Bridge běží pouze v paměti RAM počítače na němž byl spuštěn. Jelikož není propojen se souborovým systémem, nebo relačními databázemi, které by mohly zpomalovat běh Radical Bridge (vyjma logování, které může být kdykoliv vypnuto) je odezva velice rychlá i při velkém počtu připojených klientů.

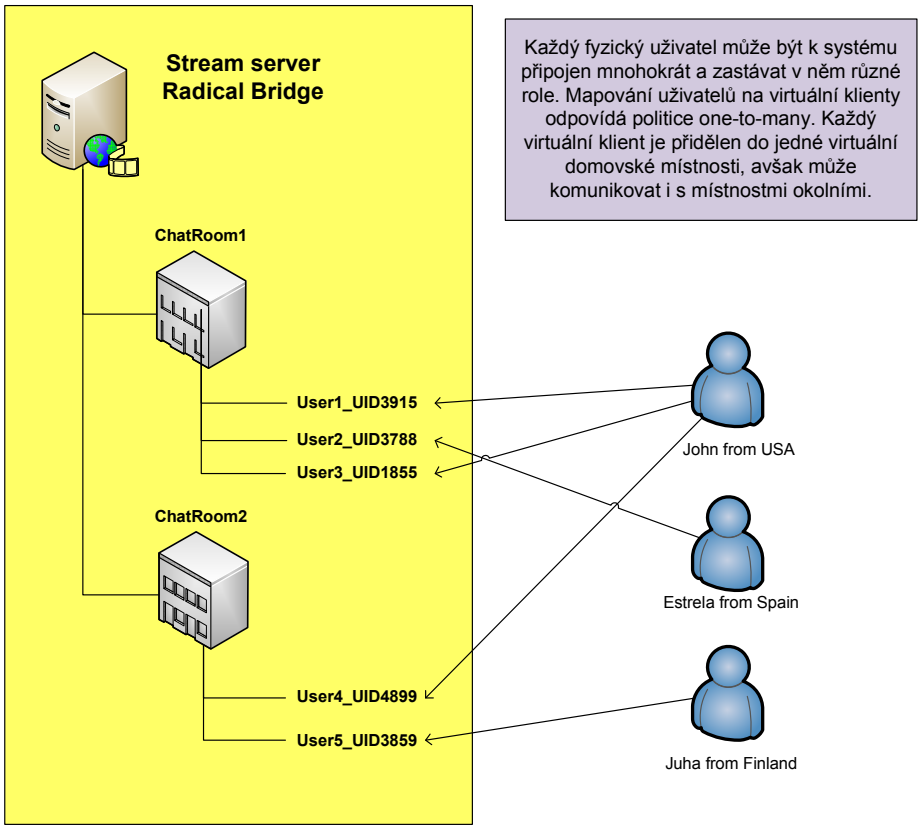
Nejvyšší řídící vrstvou celého systému je Radical Web Service. Jedná se o webovou službu implementovanou vývojářem používajícím framework Radical Chat. Radical Bridge (jádro frameworku) zde vystupuje v roli klienta webové služby Radical Web Service, která teprve musí být implementována koncovým vývojářem. Takovéto neobvyklé řešení a dalo by se i říci precedens v používání webových služeb se zprvu může zdát jako velice nešťastné, ale opak je pravdou. Právě díky této volbě je možné řídící vrstvu implementovat v jakémkoliv programovacím jazyce a framework použít na jakékoliv platformě. Pro usnadnění vývoje je webová služba předprogramována ve vybraných jazycích (PHP a C# ASP.NET) a dodána spolu s použitelným API, které značně usnadňuje tvorbu aplikací. Koncoví vývojáři si díky tomu nemusejí lámat hlavu a složitě re-implementovat celou službu podle přiloženého WSDL dokumentu. Stačí jim pouze pozměnit pár řádků kódu - vytvořit vlastní logovací systém, vzhled a pravidla aplikace.

3.2 Vnitřní logika systému

Jakožto všechny ostatní multimediální frameworky a videokonferenční systémy i Radical Chat má některá vlastní základní pravidla, která musí být dodržována.

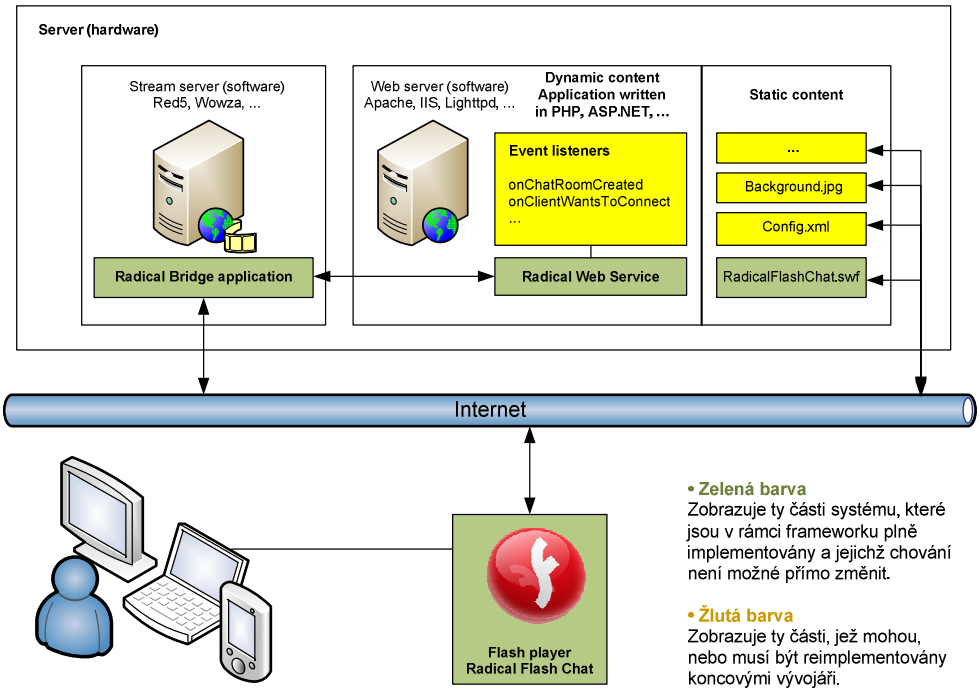
Návrh počítá s tím, že v systému existují dva nejzákladnější typy objektů: klient (uživatel používající aplikaci) a místnost (virtuální adresář klientů). Každý objekt je identifikován globálně jedinečným identifikátorem UID. Ten musí být unikátní nejen v rámci místnosti, nebo serveru, ale měl by být unikátní v rámci celého světa. Tento identifikátor musí být přidělen klientovi ještě před tím než se systémem začne pracovat. Toho je možné dosáhnout nastavením hodnoty flashvars, nebo pomocí konfiguračního souboru config.xml. Délka ani tvar identifikačního řetězce není nijak omezen. Návrh frameworku počítá s tím že logování a generování všech nezbytných UID si

zabezpečí vývojář(uživatel frameworku) sám. Koneckonců celé jádro Radical Chatu běží pouze v operační paměti a nepřímo se počítá s tím, že napojení na databáze případně adresář pomocí LDAP protokolu zabezpečí vývojář v rámci nejvyšší vrstvy Radical Web Service. Každý klient při vstupu do systému musí mít kromě UID nastavenu příslušnost k právě jedné místnosti v rámci níž může komunikovat s jinými klienty (to samozřejmě neznamená, že by nemohl navazovat spojení s jinými uživateli jiných místností). Celá struktura je velice intuitivní a ve zjednodušení vypadá nějak takto:



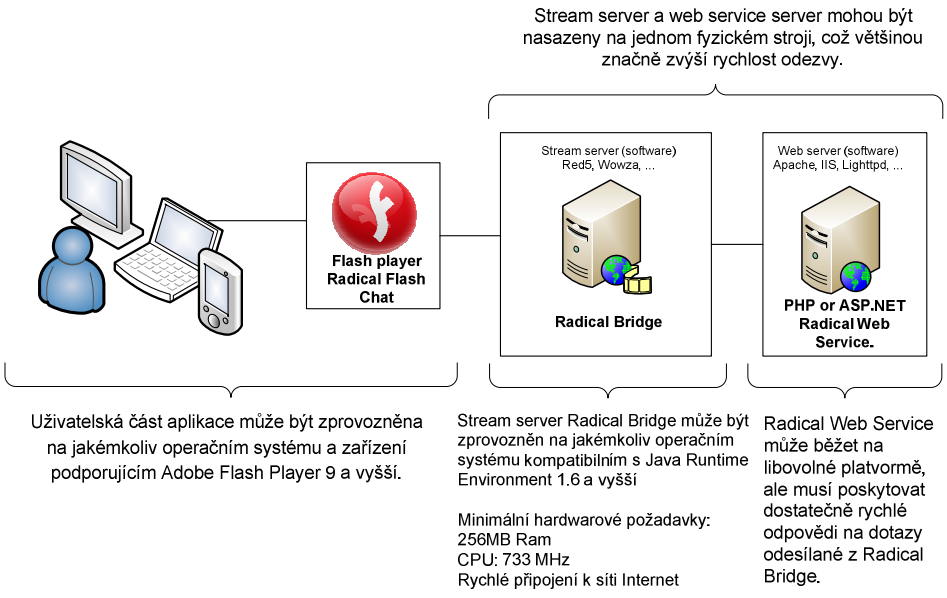
Obrázek 3 - Hierarchie základních objektů [4]

Aby byla zajištěna maximální jednoduchost použití, ale také univerzálnost celého řešení - byl po delších úvahách zvolen následující dvoustupňový model spouštění aplikací. Ihned poté co uživatel nastartuje Radical Flash Chat, tak se RFC pokusí stáhnout konfigurační soubor config.xml. Radical Flash Chat je jen malá téměř 80 kB aplikace psaná ve Flashi (ActionScript 3) obsahující virtual machine pro zpracování příkazů přicházejících ze strany serveru. Ve výchozím stavu nedělá nic a pouze zobrazí prázdnou bílou oblast na obrazovce uživatele. Teprve po stažení konfiguračního souboru jsou na plochu rozmístěny grafické prvky, nastaveny jejich funkcionalita a základní vlastnosti (jako UID klienta a místnosti do níž má být přidělen). Následně se Radical Flash Chat pokusí připojit do struktury systému a navázat spojení s Radical Bridgem.



Obrázek 4 - struktura systému Radical Chat [4]

Pokud v momentě připojení klienta neexistuje žádná místnost, která by odpovídala specifikovanému roomUID, pak ji Radical Bridge automaticky vytvoří a alokuje pro ni potřebnou paměť. Je nutné si uvědomit, že téměř celý systém a především stream server Radical Bridge běží pouze v operační paměti serveru. V momentě kdy poslední klient opustí místnost a ta se uzavře, jsou veškerá data spojená s klienty a místností samotnou nenávratně ztracena.

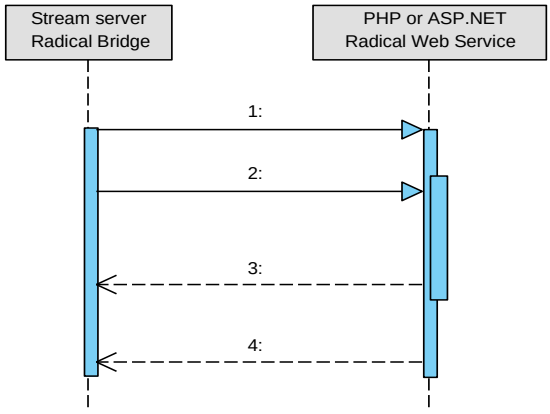


Obrázek 5 - rozložení systému [4]

Zajištění perzistence informací a případného stavu objektů je výhradně na vývojáři a vrstvě Radical Web Service. Komunikace mezi vrstvou Radical Brdige a Radical Web Service byla několikrát přebudována a její návrh nebyl v žádném případě jednoduchý.

3.3 Návrh komunikace mezi Radical Bridge a Radical Web Service

Prvotní myšlenkou celého systému bylo dát budoucím vývojářům do ruky nástroj, pomocí něhož by mohli velice rychle a relativně snadno psát kód (prakticky v libovolném jazyce) tak, aby výsledná aplikace byla výkonná a spustitelná na jakémkoliv počítači, či mobilním telefonu, uměla přehrávat nejrůznější multimediální formáty a dokonce spojovat lidi pomocí tele-mostů. Jasnou volbou pro komunikaci mezi vrstvou Radical Bridge a Radical Web Service je technologie webových služeb. Zde ale návrh naráží na tři nečekané a podstatné problémy. Webové služby jsou bezestavové (mezi jednotlivými dotazy není možné udržovat žádný relevantní kontext). Jsou synchronní - to znamená na jeden dotaz pocházející z Radical Bridge, přijde od Radical Web Service právě jedna odpověď. Během zpracovávání dotazu nemůže webová služba interaktivně komunikovat se stream serverem. A snad úplně to nejdůležitější a nejzáludnější - běh Radical Bridge je vysoce paralelizován. Což nikomu (ani webovým službám, ani vývojářům používajícím framework) neprospívá. V praxi se velice často vyskytují situace kdy se různá vlákna Radical Bridge dotazují v jeden jediný okamžik na mnoho vzájemně propojených informací služby Radical Web Service a není zaručeno pořadí v jakém budou dotazy seskupeny a zpracovány.

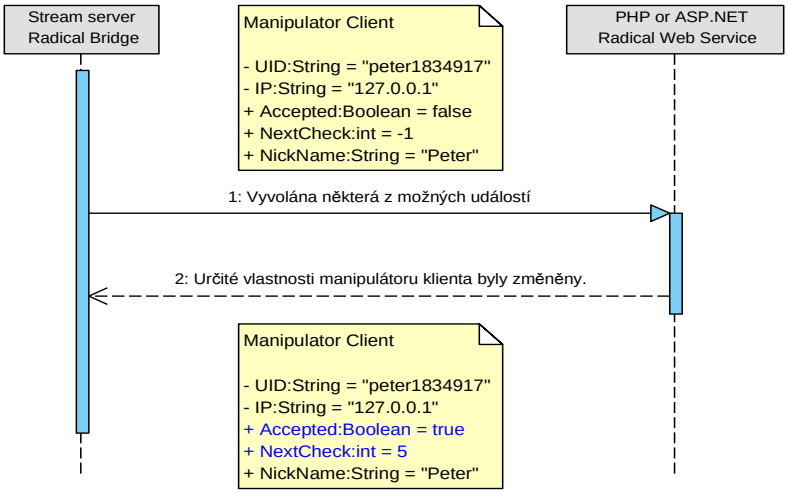


Obrázek 6 - zpracování paralelních dotazů

Návrh komunikace v nejvyšší vrstvě frameworku se snaží řešit všechny tři problémy najednou pomocí technologie virtuálních manipulátorů, zřetězení příkazů a sekvenčních zámků.

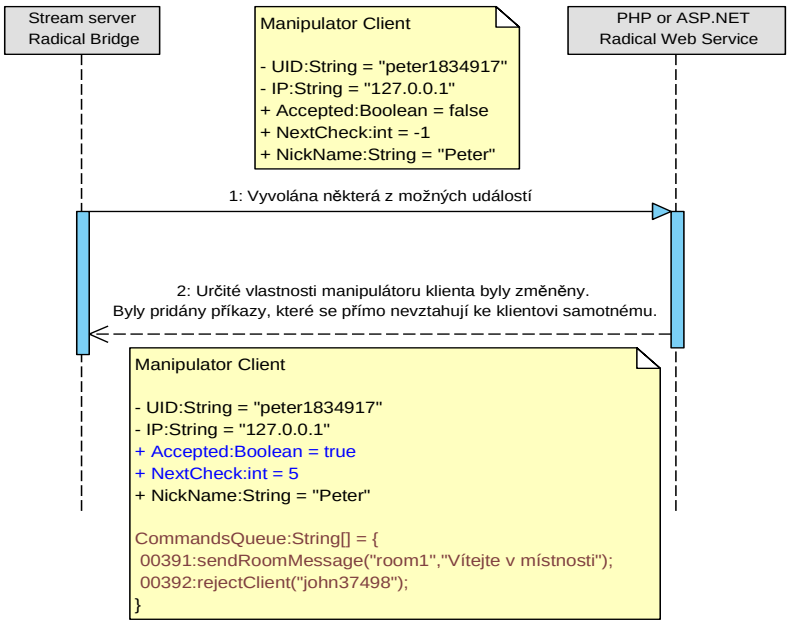
3.4 Technologie virtuálních manipulátorů

Termínem virtuální manipulátor je myšlen generalizovaný (zobecněný) objekt (definovaný množinou různorodých parametrů), který fyzicky existuje jen ve vrstvě Radical Bridge. Virtuální manipulátor umožňuje webové službě Radical Web Service měnit některé klíčové vlastnosti objektu a následně o jejich změně transparentně informuje stream server.



Obrázek 7 - ovládání systému za pomoci virtuálních manipulátorů [4]

Kromě toho ještě každý manipulátor obsahuje tzv. CommandsQueue (frontu příkazů), díky níž mohou vývojáři asynchronně měnit globální chování a nastavení stream serveru Radical Bridge, případně upravovat vlastnosti jiných objektů.



Obrázek 8 - Fronta příkazů [4]

Obrázek výše jasně ukazuje, jak je možné zabezpečit efektivní komunikaci dvou nejvyšších vrstev frameworku tak, aby webová služba uspokojivě ovládala chování celé aplikace. Přitom je běh systému maximálně rychlý, neboť je webová služba(obvykle psaná v jazyku PHP) dotazována pouze jednou za čas - v momentě kdy nastane nějaká z primárních událostí (primární události jsou popsány níže v textu).

3.5 Události zpracovávané Radical Web Service

Radical Web Service tvoří mozek celého systému a rozhoduje o všem podstatném. Pro zajištění dobré odezvy však tato vrstva nesmí být zahlcována přemírou požadavků. Na druhou stranu měla by být informována o všem podstatném co uživatelé provádějí. Dilema o tom co má být automaticky rozhodováno Radical Bridgem a co už by měl řešit kód vývojáře je jednou z nejdiskutovanějších oblastí celého frameworku. Je velice důležité umožnit vývojářům používajícím tento MMVCF zasáhnout prakticky do všeho, byť i sebeméně podstatné.

Radical Bridge se stará o automatické přehrávání multimediálních proudů dat. Dokáže obsluhovat tisíce klientů a virtuálních místností. Zabezpečuje textové chatování, nebo koordinuje přímé propojení klientů - například kvůli posílání velkých datových souborů. Všechny tyto rutiny jsou již předprogramovány v jádře Radical Bridge a jejich funkce otestována. Otázkou zůstává: kdy a pro jaké klienty dané operace použít. Na to musí odpovědět právě Radical Web Service a kód naprogramovaný vývojářem.

Radical Bridge běží teoreticky nonstop a v paměti RAM udržuje informace o všech připojených klientech a místnostech které jsou v dané chvíli aktivní. Bohužel ne vždy si ví rady jak na konkrétní situaci zareagovat. V takovém případě použije službu Radical Web Service a vyvolá jednu z množiny podporovaných událostí. V aktuální verzi Radical Chat 1.0.0.1455 je to 18 událostí, které jsou rozděleny do 4 skupin.

Cílem níže uvedeného textu je ukázat některá ze zajímavých řešení použitých při komunikaci mezi Radical Brdige a Radical Web Service, nikoliv popisovat funkce z pohledu programátora. Kompletní popis API a dokumentaci všech parametrů - vracených a přijímaných událostmi je možné nalézt na stránkách autora, nebo nejlépe přímo v dokumentaci zdrojového kódu. (Viz. přílohy).

3.5.1 Globální události

Globální události slouží jednak pro zajištění zpětné kompatibility s předešlými a budoucími verzemi Radical Chatu, ale také pro zjištění rozsahu použitelnosti Radical Web Service. Obě níže uvedené funkce jsou v Radical Web Service vyvolány jako první ihned poté co je Radical Bridge spuštěn.

- **GetAPICode** - vrací identifikační řetězec aplikačního programového rozhraní použitého na počítači, kde běží Radical Web Service.
- **GetImplementedFunctions** - funkce vracející čárkou oddělený seznam všech událostí, které chce vývojář nějakým způsobem zpracovávat. Pokud se programátor používající tento MMVCF rozhodne, že nějakou konkrétní událost vůbec zpracovávat nebude a vystačí si s výchozí reakcí, pak je vhodné - právě zde oznámit stream serveru Radical Bridge, že ke zpracování dané konkrétní události není nutné volat Radical Web Service. Tato strategie šetří spotřebovaný procesorový čas a objem přenesených dat celému systému.

3.5.2 Aplikační události

Jedná se o druhý stupeň globálních událostí oznamujících, že stream server byl úspěšně nastartován, otestoval Radical Web Service a je připraven na přijímání prvních klientů.

- **OnApplicationStarted** - aplikace byla nastartována, potřebné datové struktury byly vytvořeny a systém čeká na přihlášení prvních klientů.
- **OnApplicationStopped** - událost vyvolaná těsně pře tím než je stream server korektně vypnut, nebo po dlouhé době nečinnosti, kdy aplikace přechází do takzvaného stavu hibernace - což opět šetří prostředky systému.

3.5.3 Přístupové události

Přístupové události řeší situace, kdy se nějaký konkrétní klient snaží připojit k systému a spustit aplikaci, nebo ze systému odchází.

- **OnClientWantsToConnect** - událost je vyvolána v momentě, kdy se nějaký nový klient snaží připojit k systému, tedy ještě před tím než je do systému vpuštěn. Již je známa jeho IP adresa, unikátní ID a ID místnosti do níž chce vstoupit.
  Na této funkci je možné názorně demonstrovat jedno zajímavé a na první pohled ne příliš logické chování Radical Chatu. Pokud řetězec "onClientWantsToConnect" není uveden v seznamu zpracovávaných událostí (funkce "GetImplementedFunctions"), pak je klient (uživatel) automaticky přijat - neboť takové je výchozí chování "onClientWantsToConnect". Pokud je ale událost zpracovávána kódem vývojáře a ten v ní explicitně nenastaví klientův příznak Accepted, anebo funkce kompletně selže, pak je uživatel, který se snaží aplikaci spustit okamžitě odmítnut a nemůže se systémem dále pracovat. Proč tomu tak je? Praxe ukázala, že vývojáři (především PHP vývojáři) používající framework Radical Chat dělají poměrně často syntaktické a sémantické chyby. Může se stát, že v momentě zpracování "onClientWantsToConnect" právě takováto chyba nastane. Proto je voleno nejrestriktivnější možné opatření, aby se do systému nedostaly nepovolané osoby. Má-li kód programátora zpracovávat nějakou konkrétní událost a vyskytne-li se při zpracování chyba - tzn. Webová služba selže a neodpoví, pak je voleno nejpřísnější možné řešení, což zabrání možným nekonzistencím. V tomto případě odmítnutí klienta vstupujícího do systému.
- **OnClientConnectionWasAccepted** - událost nastane, jakmile je uživatel přijat do systému, proběhne úvodní handshake, synchronizace skinu a chování aplikace s příslušnou chatovací místností.
- **OnClientConnectionWasRejected** - nastane v případě, že bylo připojení uživatele odmítnuto funkcí "onClientWantsToConnect".
- **OnClientDisconnected** - nastane v případě, kdy klient aplikaci ukončí, nebo mu selže připojení k internetu, nebo například vypadne proud v domácnosti.

3.5.4 Události pro kontrolu přehrávání multimediálního obsahu

Tyto události jsou vyvolány pokaždé, kdy se určitý klient pokusí zahájit přehrávání audia, či videa, nebo se pokusí zahájit živé streamování. Slouží především pro kontrolu - co který klient může přehrávat a publikovat.

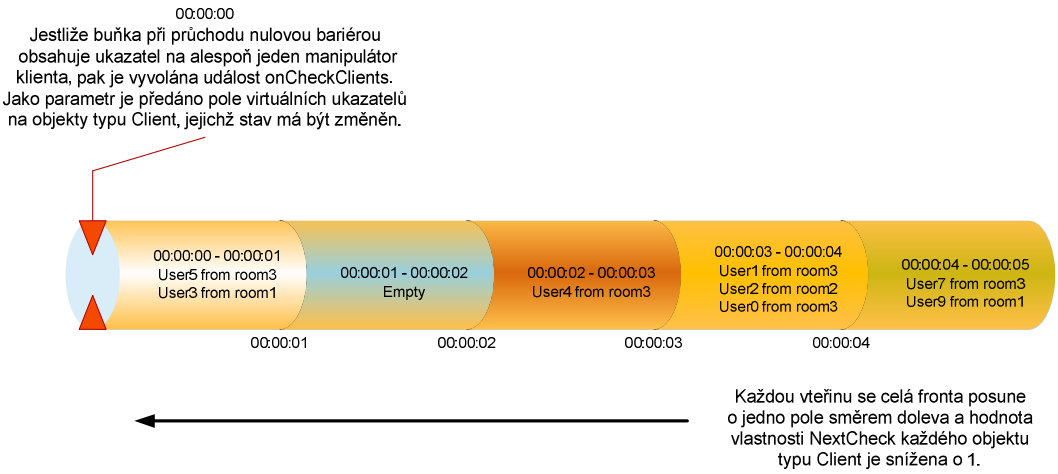
- **OnClientWantsToStartStreaming** - událost je vyvolána těsně před tím než stream server přijme stream publikovaný klientem. Většinou se jedná o živé video nahrávané z webové kamery uživatele.
- **OnClientStoppedStreaming** - oznamuje, že klient přestal publikovat živý stream.
- **OnClientWantsToPlayStream -** je vyvolána v momentě, kdy se určitý klient pokusí přehrát audio, nebo video (ať již uložené v souboru, nebo živé).
- **OnClientStoppedPlayingStream** - oznamuje, že konkrétní klient přestal přehrávat určitý stream.

3.5.5 Události vnitřního filtrování

Tyto události bývají spouštěny pouze v případech, kdy je to u konkrétního klienta explicitně vyžadováno. Například: filtrování obsahu textových zpráv. K tomu, aby bylo možné některou z následujících funkcí použít - musí být patřičně nastaveny určité přepínače manipulátoru klienta (většinou se tak děje v události onClientWantsToConnect).

- **OnClientSendTextMessage** - je-li přepínač dispatchEventIfClientSendTextMessage nastaven na hodnotu true, pak je každá textová zpráva, kterou daný uživatel odešle zkontrolována metodou OnClientSendTextMessage (vrstvy Radical Web Service).
- **OnCheckClients** - Tato funkce je spouštěna v periodických intervalech a funguje podobně, jako časovač, nebo naplánovaná úloha. Její použití je rozmanité a v praxi se ho hodně využívá. Hlavní myšlenka vychází ze základního požadavku vývojářů. Radical Bridge běží bez zastavení v nekonečném cyklu, zatímco Radical Web Service je dotazován pouze jednou za čas klasickou metodou request-response. Běh kódu Radical Web Service od spuštění po ukončení trvá řádově milisekundy. PHP nebo ASP.NET aplikace nemůže běžet bez ustání. Nemůže být propojená s několika uživateli najednou, posílat jim nová data a příkazy podobně jako stream server. Drtivá většina webových serverů takové chování nedovoluje. V některých případech je ale nezbytně nutné periodicky kontrolovat a měnit stav klienta, upravovat vzhled nebo chování aplikace. Právě proto má každý virtuální manipulátor Client vlastnost NextCheck, který umožňuje nastavit vteřinový interval pro naplánování další interakce vrstvy Radical Bridge s vrstvou Radical Web Service a navodit tak dojem, že Radical Web Service běží neustále.

Bohužel může nastat i situace, kdy se v jednom vteřinovém okně vyskytne více požadavků na vykonání různorodých akcí onCheckClients. To by mohlo způsobovat značné zpomalení běhu serveru, a proto jsou všechny události, které mají nastat v daném vteřinovém okně sloučeny do jedné kumulativní události a předány jako parametr funkce onCheckClients.



Obrázek 9 - onCheckClients

3.5.6 Události spojené s virtuálním objektem místností.

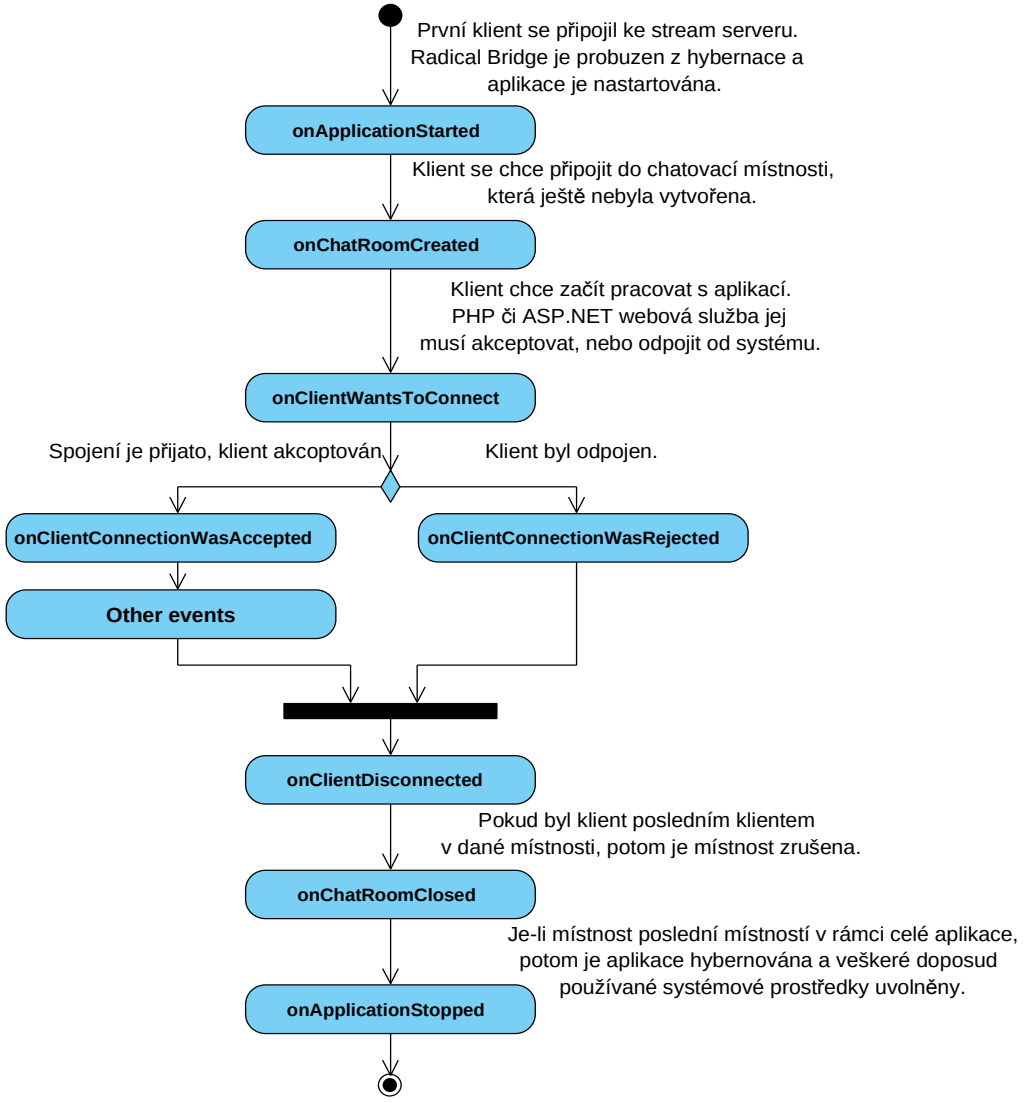
Následující tři události jsou spojeny s životem objektu ChatRoom.

- **OnChatRoomCreated** - je spuštěn v momentě, kdy je vytvořena nová virtuální místnost, těsně před tím než do ní vstoupí první klient.
- **OnChatRoomClosed** - událost je vyvolána těsně poté, co místnost opustí poslední klient.
- **OnCheckChatRooms** - událost je svým chováním velice podobná “onCheckClients”. Pro její spuštění je nutné upravit vlastnost NextCheck dané místnosti.

3.5.7 Univerzální události

Všechny univerzální události jsou zpracovávány funkcí onUniversalCall

- **OnUniversalCall** - je událost používaná především k přímému propojení grafického uživatelského prostředí aplikace a služby Radical Web Service. Typicky je vyvolána poté co uživatel stiskne v aplikaci určité předvolené tlačítko, nebo může být použita k synchronizaci grafických uživatelských rozhraní více navzájem propojených klientů.



Obrázek 10 - události

3.6 Programování v asynchronním vysoce paralelizovaném prostředí

Jak bylo řečeno v kapitole virtuálních manipulátorů - Webová služba umožňuje vývojářům ovládat takřka celý systém a navozuje v nich mylný dojem, že pracují s aplikací v reálném čase. Z jejich pohledu se programování v Radical Chatu jeví jako jednoduché přímočaré a synchronní. Tak tomu bohužel není. Pokaždé, když je ve vrstvě Radical Web Service použita některá ze základních API funkcí multimediálního videokonferenčního frameworku - je volání této funkce přeloženo do pseudokódu a uloženo do fronty příkazů, která se následně odešle jako výsledek dotazu na webovou službu. Fronta příkazů je sériově zpracována až teprve poté co Radical Bridge přijme odpověď od Radical Web Service. Takovéto řešení má mnoho nečekaných výhod, ale i skrytých problémů, které je třeba řešit.

3.7 Výhody programování v asynchronním paralelizovaném prostředí

Představme si následující futuristickou webovou aplikaci, ve které může chování jednoho uživatele v reálném čase ovlivňovat vzhled a funkcionalitu dané aplikace zobrazené mnoha jinými uživateli. Nebo jiný mnohem komplikovanější scénář. Některý z uživatelů odešle textovou sms zprávu - ta prostřednictvím brány mobilního operátora dorazí na server klienta, spustí PHP nebo C# skript, který okamžitě odpojí několik uživatelů od systému, u jiných změní vzhled aplikace a zahájí přehrávání videa. Něco takového je samozřejmě zcela nemožné udělat prostřednictvím klasického request-response modelu používaného ve webových službách, nebo webových aplikací psaných v PHP či ASP.NET. Zmiňovanou situaci neřeší ani dnes tolik diskutovaný AJAX (technologie pro asynchronní výměnu údajů mezi webovým prohlížečem a webovým serverem) - stále se jedná o request-response mechanizmus. Na jeden dotaz, jedna odpověď.

Tento model byl zatím překonán pouze v několika málo technologiích. Mezi ně patří například Microsoft DirectX (konkrétně knihovna DirectConnect), WebSockets (HTML5) nebo Adobe Flash.

V jazyce ActionScript 3.0 a přehrávači Adobe Flash Player verze 9.0 a vyšší jsou prakticky všechny události a příkazy sloužící pro komunikaci se serverem asynchronní. Je zde ale jedna podstatná maličkost, která zásadně odlišuje chování Adobe Flash Playeru a zároveň i Radical Flash Chatu od klasických aplikací psaných v HTML4, používajících technologii SilverLight, nebo Java Appletů. Adobe Flash Player poté co otevře rtmp spojení se stream serverem a získá požadovaná data takovéto spojení neuzavírá a ponechává ho otevřené po celou dobu běhu flashové aplikace (nebo dokud jej programátor sám explicitně neuzavře). Díky tomuto promyšlenému chování mohou v Adobe Flash Playeru a Radical Flash Chatu nastat dva nové typy událostí, které klasický protokol HTTP nepodporuje.

1. Response without request - aplikace může obdržet nová data, i když o ně vůbec nepožádala. Toho využívá například interní třída SharedObject (zabudovaná přímo v základním API pro Adobe Flash Player 8.0 a vyšší), která synchronizuje informace napříč složitým systémem různě propojených klientů.
2. Request without response - stream server může získat od klienta zprávu, nebo oznámení na které není nutné odpovídat.

Jelikož je webový server odcloněn od přímé komunikace s jednotlivými klienty (mezi vrstvou Radical Web Service a Radical Flash Chat stojí stream server Radical Bridge) je možné poměrně jednoduchým způsobem umožnit PHP, ASP.NET,… vývojářům interagovat se serverem. Ten může vyvolávat různé akce a ovlivňovat chování několika set připojených klientů jediným příkazem. Vše je zabezpečeno právě díky frontě pseudopříkazů (CommandsQueue) vyprodukované kódem vývojáře po vyvolání některé z primárních událostí vrstvy Radical Web Service.

3.8 Nevýhody programování v asynchronním paralelizovaném prostředí

Předávání příkazů asynchronní metodou, například pomocí několikrát zmiňované proměnné CommandsQueue a jejich následné (opožděné) vykonání není vůbec žádný problém. Dokonce i rychlost zpracování je výrazně vyšší, než kdyby byly příkazy emulovány či jinak spouštěny přímo web serverem. Jednou z drobných překážek může být fakt, že pokud se na závěr PHP nebo ASP.NET skriptu v Radical Web Service vyskytne nečekaná chyba, pak je celá sekvence pseudopříkazů zahozena. Na druhou stranu - každý vývojář musí počítat s tím, že událost (v Radical Web Service) je Radical Bridgem chápána jako jednolitý celek ne nepodobný zpracovávání transakcí v relačních databázích.

Veliký problém ale nastává v momentě, kdy se do hry vmísí paralelizmus (Viz. obrázek 6 - zpracování paralelních dotazů). Při rozplétání problémů spojených s paralelizmem může dojít k zacyklení (deadlocku), vykonání příkazů ve špatném pořadí, nebo neúměrnému zpomalení vlivem “vyčkávání na synchronizační bod”. Proto byl pseudokód používaný frontou příkazů navržen tak, aby jednotlivé funkce byly pokud možno zcela nezávislé na ostatních (tím je řešen výhradně deadlock). Dále pak má každý asynchronně vykonávaný příkaz použitý v Radical Web Service přiřazenu časovou značku. Jde o sériové pořadové číslo označující, kdy má být příkaz v sekvenci vykonán. Toto sériové číslo se používá především v případech “úplné synchronizace” - například problém dynamického skinování. Většinou si ale Radical Chat vystačí s částečnou synchronizací, která dává v drtivé většině případů velice dobré výsledky za cenu občasného zpřeházení řádků programového kódu. Níže je uveden příklad řešený částečnou synchronizací.

```
function _onClientWantsToConnect(&$client)
{
  Logger::func("onClientWantsToConnect");
  //accept client's connection
  $client->AcceptConnection();
  $client->SendPrivateMessage("Admin", $client->getUID(),
                              "000000", "Welcome in the room.");
  $client->SendRoomMessage("Admin", "000000",
                           "User \"" . $client->getNickName() .
                          "\" has entered the room.");
  $client->SendPrivateMessage("Admin", $client->getUID(), "000000",
                 "Please do not use unpolite words in this chat.");
  $client->SendRoomMessage("Admin", "000000",
                 "Remember this room will be closed at 9pm.");
}
```

Otázka zní, co se stane v případě, že se do systému přihlásí dva uživatelé současně v jeden okamžik. Jelikož je problém textového dopisování řešen pouze částečnou synchronizací a navíc jsou veřejné a privátní zprávy putující k jednotlivým uživatelům vzájemně mixovány - může výsledek v aplikaci uživatele 1 a 2 vypadat následovně:

Uživatel 1:

```
Admin: User "User2" has entered the room.
Admin: Remember this room will be closed at 9pm.
Admin: User "User1" has entered the room.
Admin: Remember this room will be closed at 9pm.
Admin>> User1: Welcome in the room.
Admin>> User1: Please do not use unpolite words in this chat.
```

Uživatel 2:

```
Admin: User "User2" has entered the room.
Admin: Remember this room will be closed at 9pm.
Admin>> User2: Welcome in the room.
Admin>> User2: Please do not use unpolite words in this chat.
Admin: User "User1" has entered the room.
Admin: Remember this room will be closed at 9pm.
```

3.9 Konvertor video formátů a video střižna

Každý solidní multimediální framework nabízí vývojářům sadu funkcí pro úpravu a převod mezi jednotlivými multimediálními formáty. U Radical Chatu však nastává drobná komplikace. Celý systém, tak jak byl doposud zmiňován, je možné nasadit na prakticky libovolný webhosting. Stačí pokud daný poskytovatel nainstaloval na server Java Runtime Environment verze 1.6, některý z mnoha podporovaných webových serverů (jako třeba Lighttpd, Apache, IIS, ...) a interpret jazyka PHP, nebo C# (.NET framework). V takovém případě bude vše bezchybně fungovat bez ohledu na použitý operační systém, nebo hardware. Video konvertor je na rozdíl od celého zbytku systému postaven na knihovnách FFMPEG a rozšířeních použitelných pouze pod operačním systémem Linux. Existuje sice i verze FFMPEGu pro Windows a Mac OS nicméně rozšiřující moduly a API nejsou plně kompatibilní. Z toho důvodu je “Radical Chat Video Convertor (RCVC)” dodáván k frameworku jako volitelné rozšíření. Množina použitelných funkcí je závislá na operačním systému a výkon na hardwaru počítače. RCVC poskytuje v jádru ty nejdůležitější funkce, které vývojáři potřebují ke zpřístupnění statického multimediálního obsahu svým klientům. Za všechny stačí zmínit

- convertVideoToMP4 - převede video zaznamenané téměř libovolným zařízením v libovolném video kodeku do MP4 (H.264) formátu používaném technologií Adobe Flash Player a Radical Flash Chat. Přitom má vývojář možnost ovlivnit kódování, rozlišení a dokonce vložit do snímku vodoznak.
- convertVideoToFLV - převede libovolné video do staršího formátu FLV.
- createThumbnail - vytvoří náhled videa v určité vteřině, či milisekundě a uloží ho do JPG nebo PNG.

Jelikož video konvertor pouze zapouzdřuje vnitřní funkce knihovny FFMPEG a nepřináší žádné objevné poznatky nebude se jím text této práce nadále zabývat. Rozšiřující informace je možné nalézt v dokumentaci výrobce, nebo v popisu zdrojových kódů rozšíření Radical Chat Video Convertor.

4 Grafické uživatelské rozhraní

Tato kapitola se zabývá návrhem grafického uživatelského rozhraní v univerzálním multimediálním videokonferenčním frameworku Radical Chat. První část je věnována statickému skinování a statickému grafickému kontextu aplikací. Druhá část popisuje návrh a implementaci jednotlivých grafických komponent.

4.1 Statický kontext aplikace

Pojmem statický kontext aplikace (psané ve frameworku Radical Chat) je myšlen vzhled a základní chování aplikace spuštěné na počítači koncového uživatele ještě před tím, než se Radical Flash Chat připojí ke stream serveru Radical Bridge. Statický kontext se používá při startu aplikace na počítači klienta a funguje obdobně jako bootovací sekvence při spouštění PC, nebo jako zavaděč operačního systému. Ke zjištění a nastavení statického kontextu aplikace je použit výhradně protokol HTTP, případně zabezpečený HTTPS. Tento kontext slouží především k rozmístění aktivních prvků a vizuálních komponent na ploše výchozího formuláře, ale také identifikuje adresu stream serveru, ke kterému se má Radical Flash Chat připojit.

Grafické uživatelské rozhraní ve frameworku Radical Chat je inspirováno základní myšlenkou formátu XHTML a technologií Windows Presentation Foundation, ale přináší také spoustu nových moderních úvah a inovativních postupů. Díky tomu, že grafické uživatelské prostředí je ve většině případů definováno pouze statickým kontextem – je tím elegantně oddělen vzhled aplikace od jejího složitějšího vnitřního chování. Designéři mohou navrhovat skiny videoplayerů, videokonferencí, nebo videochatů bez asistence programátorů. Léty prověřená softwarová architektura MVC je tedy přítomna i v Radical Chatu.

4.2 Radical Chat Skinning Language

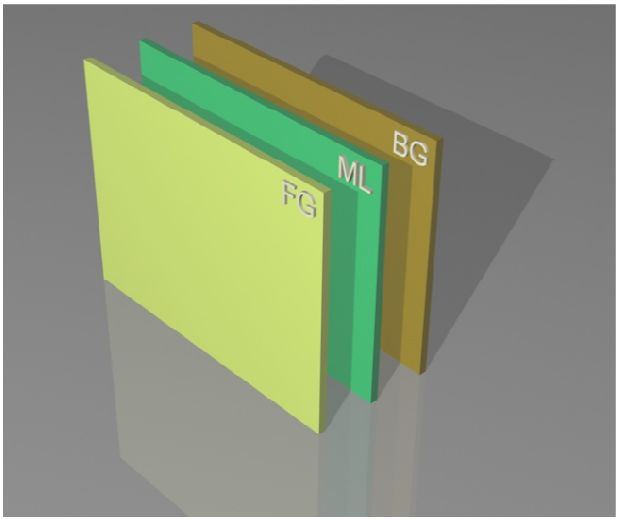
Ještě před zahájením práce na frameworku se vývojáři dlouze a pozorně zamysleli nad tím, jaké nástroje dát do ruky budoucím návrhářům vzhledu aplikací. Prohlédli si všechny moderní technologie od plánovaného HTML5 přes WPF až po FLEX a zjistili, že žádná z nich není dostatečně vyhovující a přiměřeně jednoduchá. Vznikl proto nový jazyk RCSL (Radical Chat Skinning Language) který je inspirován tím nejlepším z výše zmíněných technologií, ale jde svou vlastní cestou.

RCSL je o své podstatě XML dokument, který podléhá určitým speciálním pravidlům a zákonitostem. Obsahuje řadu vrstev (v současné době pouze 3), ty mohou obsahovat grafické

komponenty, nebo další vrstvy podle potřeby. Díky tomu, že je jazyk RCSL textovým jazykem postaveným na XML – je možné se jej velice rychle naučit (v řádu minut) a vyvíjet kód na libovolné platformě. Většina moderních integrovaných vývojových prostředí (NetBeans, Zend, Microsoft Visual Studio) obsahuje kontrolu XML syntaxe a snižuje tak pravděpodobnost výskytu chyby. RCSL skripty je možné psát úplně všude. Dokonce i na mobilních telefonech.

4.3 Grafické vrstvy

Autor frameworku se při návrhu API pro designéry budoucích multimediálních aplikací zaměřili především na jednoduchost a přímou cestu vývoje. Mezi základní stavební kameny patří takzvané vrstvy, nebo též kontejnery grafických prvků. U většiny programovacích jazyků určených pro tvorbu vzhledů aplikací jako je HTML, TCL/TK existuje něco jako výchozí plátno, na něž je možné kreslit. U Radical Chatu je nutné nejprve v rámci výchozího plátna vytvořit vrstvu, která zastává roli kontejneru grafických prvků. V aktuální verzi 1.0.0 existují pouze tři základní vrstvy BG (Background layer), ML(Main layer) a FG(Foreground layer). Kromě toho si ještě každý grafický prvek vytváří svou vlastní podvrstvu. Proč se jedná o právě tyto tři? Vrstva ML (Main Layer) může obsahovat pouze takzvané toxické komponenty. Jedná se o aktivní grafické prvky, které se mohou v rámci celé aplikace vyskytovat maximálně jednou a to jen a pouze ve vrstvě ML.



Obrázek 11 - grafické vrstvy

Toxické komponenty není možné vytvářet dynamicky, nebo je dynamicky skinovat (musí být nastaveny ve statickém kontextu), přesto tvoří důležitou část každé multimediální videokonferenční aplikace. O toxických komponentách bude pojednáno později v textu. Vrstvy BG a FG mohou na rozdíl od ML obsahovat standardní grafické komponenty v neomezeném počtu.

4.4 SkinObject

Ještě předtím, než byly vytvořeny vůbec první grafické komponenty a ovládací prvky pro univerzální multimediální framework vznikla abstraktní třída SkinObject. SkinObject je prapředek všech vizuálních prvků, které dnes mohou vývojáři v Radical Chatu použít. Sdružuje 5 nejdůležitějších vlastností společných všem komponentám a dává vývojářům do ruky mocný nástroj s jehož pomocí jsou schopni vytvářet hardwarově akcelerované efekty. Tento nápad vznikl již v počátcích vývoje univerzálního multimediálního videokonferenčního frameworku a reaguje na chyby v návrhu jazyka HTML. Ten ani po rozšíření o kaskádové styly nedokáže řešit některé základní požadavky grafiků.

Obrázek s třídami grafických prvků

SkinObject je abstraktní grafickou komponentou od níž jsou odvozeny všechny další vizuální prvky (Viz. obrázek výše). Není možné jej přímo instanciovat, ale je možné SkinObject explicitně přetypovat (konkretizovat).

Vše je nejlépe vysvětleno na následujícím úseku zdrojového kódu:

```
<RadicalChat>
...
  <Skin>
  ...
    <BG>
      <SkinObject>
        <!--
          Type určuje která z universálních a nebo specializovaných
          komponent bude instanciována. Type je povinný parametr.
        -->
        <Type></Type>
        <!-- Název objektu – povinný parametr. -->
        <Name>Panel1</Name>
        <!-- Viditelnost (volitelný parametr, výchozí hodnota true). -->
        <Visible>true</Visible>
        <!-- Relativní pozice – povinná sekce -->
        <Position>
          <Left>0%+10+{AnotherObject.Right}</Left>
          <Top>0%+10</Top>
          <Right>100%-10, min=400</Right>
          <Bottom>100%-10, min=300, max=900</Bottom>
        </Position>
        <!-- Grafické efekty týkající se objektu – volitelná sekce -->
        <LookAndFeel>
          <!-- Průhlednost – číslo mezi 0 a 1. -->
          <AlphaBlend>1</AlphaBlend>
          <!--
            Mód zobrazení - podobně jako v Adobe Photoshopu.
            Povolené hodnoty: normal, multiply, etc...
```

```
          -->
          <BlendMode>normal</BlendMode>
          <!-- Barevná transformace -->
          <ColorTransform>
            <Color redMultiplier="0.1" greenMultiplier="0.4"
                   blueMultiplier="1" alphaMultiplier="1"
                   redOffset="128" greenOffset="128"
                   blueOffset="128" alphaOffset="0" />
          </ColorTransform>
        </LookAndFeel>
      </SkinObject>
      ...
    </BG>
   ...
  </Skin>
</RadicalChat>
```

XML element <Skin> definuje grafické uživatelské rozhraní aplikace. Uvnitř se mohou nacházet dříve zmíněné vrstvy <BG>, <ML>, <FG>. Konečně teprve uvnitř vrstev se mohou vyskytovat vizuální grafické komponenty odvozené od <SkinObject>.

4.5 Parametry abstraktního elementu SkinObject

Smysl většiny parametrů je pochopitelný už podle názvu XML tagu. Přesto je nutné objasnit některé ne příliš zřejmé zákonitosti, které byly při vytváření jazyka RCSL a objektu SkinObject zavedeny. Mezi tři povinné vlastnosti, které musí být v těle SkinObjectu explicitně uvedeny patří typ - <Type>, název - <Name>, a pozice - <Position>. Parametr type oznamuje konkretizaci SkinObjectu na některý existující grafický prvek (např. BasicPanel, BasicButton, VideoPlayer, …). Name je jedinečný identifikátor daného grafického vizuálního prvku. Svou funkcí zastává podobný význam jako parametry “name” a “id” u formulářových prvků v HTML. S <Type> a <Name> se pojí jedna mimořádná vazba o níž by měli vývojáři vědět.

Při návrhu frameworku byl kladen důraz na efektivitu a rychlost. Pokaždé, když vývojář vytvoří nějaký grafický prvek (ať již ze statického, nebo dynamického kontextu) tvorba datové struktury pro daný objekt zabere určitý čas. Je paradoxní, že když je poté tentýž objekt dynamicky vymazán a na jeho místo vytvořen nový objekt jiného typu – zabere Adobe Flash Playeru uvolnění datové struktury výrazně více času než vytváření nové. Pokud by vývojář dynamicky vytvářel objekty stejného jména, ale různých typů zabíralo by uvolňování a realokování datových struktur, takovou spoustu času, že by aplikace byla prakticky nepoužitelná. Na druhou stranu praxe ukázala, že vývojáři rádi mění dynamický vzhled aplikace a to někdy již s frekvencí vteřinových intervalů. Proto bylo zavedeno závazné pravidlo, které značně ulehčí práci Radical Flash Chatu a zefektivní vývoj. Pokaždé, když je na ploše aplikace vytvořen nějaký grafický prvek určitého jména

<Name>ABCD</Name> je vývojář povinen pro případ updatu, nebo dynamické destrukce a následného znovuvytvoření objektu se stejným jménem zachovat původní typ <Type> jaký byl uveden v případě prvního vytvoření objektu s názvem <Name>. Význam tohoto pravidla bude více objasněn až v kapitole Dynamický kontext aplikace.

Podsekce <LookAndFeel> je volitelná dává návrhářům grafiky možnost použít rozličné pixelshadery, upravit alfa kanál (průhlednost objektů) a různým jiným způsobem nastavit mixování barev. Příští verze počítá i se zavedením efektů rozmazání, vržený stín a vyzařování.

4.6 Pozicování vizuálních komponent

V každém aplikačním programovém rozhraní, které umožňuje vytvářet grafická uživatelská rozhraní existuje poziční systém pro umísťování grafických komponent. Ve Windows 32 API je to klasické čtyřčíslí left, top, width, height, které se vztahuje relativně ke kontejneru komponent v němž je vizuální prvek umístěn. V jazyku HTML se některé grafické komponenty nedají umísťovat vůbec a web designéři si musejí pomáhat tabulkami (tag <table>), které teprve rozdělí plátno do požadované mřížky. Pro HTML byl vydán pomocný popisný jazyk CSS, jehož hlavním cílem je pomoci oddělit vizuální stránku dokumentu od obsahové. Bohužel, ani poziční systém CSS nesplňuje všechny moderní požadavky designérů. Navíc je v každém internetovém prohlížeči implementován trochu jinak což značně stěžuje práci vývojářům a původní určení jazyka se tak míjí účinkem.

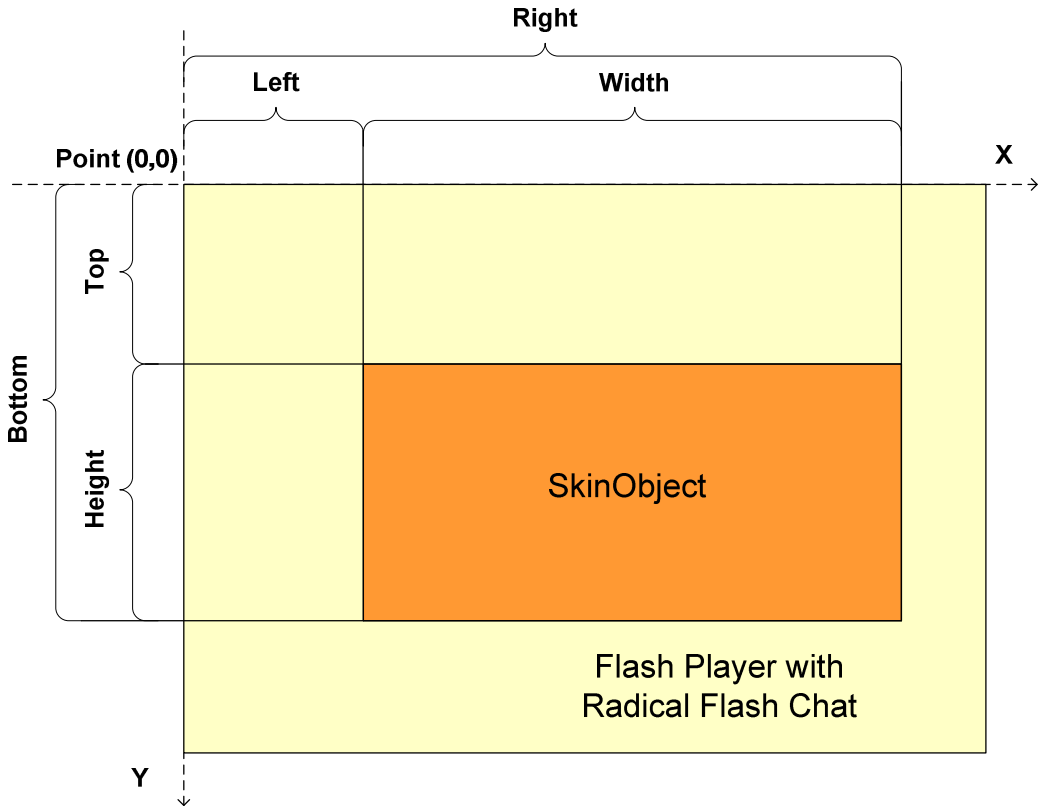
Radical Chat přichází s novou myšlenkou zvanou relativní poziční systém (RPS). Relativní poziční systém je zpětně kompatibilní s původním pozicováním v HTML, nicméně umožňuje mnohem, mnohem víc.

Jelikož je Radical Flash Chat (nejnižší vrstva multimediálního frameworku Radical Chat) implementován v ActionScriptu a spouštěn v Adobe Flash Playeru – odpadá problém testovat aplikace v jednotlivých prohlížečích. Vzhled a chování bude vždy stejné. Dalo by se očekávat, že i rozměry okna pro konkrétní finální aplikace budou vždy stejné, ale bohužel není to pravda. Radical Chat, co by multimediální framework, dává vývojářům prostor pro použití tzv. Full Screen módu (neboli zobrazení přes celou obrazovku) a jelikož různí uživatelé mohou mít různá rozlišení obrazovek objevuje se problém pozicování komponent. Radical Chat je ale kromě webového prohlížeče možné použít I jako samostatnou (stand alone) aplikaci, která je tvořena libovolně roztažitelnými okny. Proto musí univerzální multimediální framework a jazyk RCSL obsahovat efektivní mechanizmus pro relativní pozicování vizuálních komponent.

4.6.1 Relativní poziční systém

Relativní poziční systém přichází s unikátní metodou, jak umísťovat grafické prvky na plátně, nebo v rámci kontejnerů (vrstev). Byl navržen tak, aby splňoval nároky na jednoduchost, efektivitu a rychlost zpracování kódu. I když se použití nemusí zdát některým návrhářům na první pohled zřejmé, ve skutečnosti je velice jednoduché a dá se na něj velice rychle navyknout.

Základní myšlenka vychází z toho, že jakýkoliv obdélníkový grafický objekt je možné ve scéně umístit pomocí čtyř hodnot ze šesti možných (Viz. obrázek níže).



Obrázek 12 - Relativní poziční systém

Je zcela zřejmé, že z trojice hodnot Left, Right, Width stačí zadat pouze dvě a třetí může být dopočtena automaticky. Totéž platí I pro trojici Top, Bottom, Height.

$$\text{Width} = \text{Right} - \text{Left}$$

$$\text{Height} = \text{Bottom} - \text{top}$$

Na tom by nebylo nic až tak nepochopitelného, kdyby ovšem parametry Left, Right, Width, Top, Bottom, Height nemohly nabývat smíšených absolutních i relativních hodnot zároveň. Mezi

absolutní se řadí například pixely (obrazové body), palce, centimetry, mezi relativní například procenta. Navíc výše zmíněné hodnoty mohou obsahovat i odkazy na pozice jiných objektů a omezení max, min. Níže je pro názornost uvedena hodnota Left nějakého objektu Panel2.

```
<Left>85% - 50 - 5% + {Panel1.Left},min = 10% + 70,max = 500 + 30</ Left>
```

Ve výrazu je možné používat znaménka plus a mínus, nikoli však krát a děleno. Omezení min a max nejsou povinná. V případě, že min není uvedeno je automaticky nastaveno na -1 000 000 000 pixelů, v případě, že max není uvedeno je automaticky nastaveno na 1 000 000 000 pixelů.

Je jasné, že výše uvedený výraz není triviální, avšak je možné je upravit do formy kdy výpočet tohoto a jakéhokoliv jiného podobného výrazu triviální bude a bude vždy vyhodnocen v konstantním čase.

Postup výpočtu:

Během vytváření komponent jsou výrazy zjednodušeny a předpřipraveny pro rychlé výpočty. To se děje pouze jednou při spouštění aplikace. Každý parametr pozice je zjednodušen na tři vektory ($Left_{Base}$, $Left_{Min}$, $Left_{Max}$) tvořící matici následujícího tvaru.

$$\begin{matrix} \text{Left}_{\text{Base}} \\ \text{Left}_{\text{Min}} \\ \text{Left}_{\text{Max}} \end{matrix} \begin{bmatrix} \text{Pixely} & \text{Procenta} & \text{Pixely(odkaz na pozici jiného objektu A)} \\ \text{Pixely} & \text{Procenta} & \text{Pixely(odkaz na pozici jiného objektu B)} \\ \text{Pixely} & \text{Procenta} & \text{Pixely(odkaz na pozici jiného objektu C)} \end{bmatrix}$$

Poslední sloupcový vektor zvaný “Pixely(odkaz na pozici jiného objektu X)” reprezentuje jeden link na maximálně tři objekty A, B, C, avšak matice může být rozšířena o další sloupcový vektor s odkazem na objekty D, E a F. To umožní návrhářům přidat do výrazu více odkazů na pozice jiných komponent a celý systém tak mnohem efektivněji propojit. Přepočtená matice, která je uvedena níže popisuje pouze hodnotu Left. Analogicky je třeba vypočíst i matice pro zbylé tři hodnoty jednoznačně určující pozici vizuálního prvku.

Matice pro parametr Left

$$\begin{bmatrix} -50 & 80 & \&\text{Panel1.Left} \\ 70 & 10 & 0 \\ 530 & 0 & 0 \end{bmatrix}$$

Nyní už je vše velice jednoduché. K výpočtu konkrétní hodnoty Left (udávané v pixelech) je zapotřebí tuto matici zprava znásobit sloupcovým vektorem $\left\lfloor 1, \frac{\text{Layer.Width}}{100}, 1 \right\rfloor$.

Uvažujme, že aktuální rozlišení okna aplikace je 800x600 pixelů a pozice objektu Panel1 již prošla výpočtem. Hodnota Panel1.Left se rovná 100 pixelů. V tom případě bude výpočet vypadat takto:

$$\begin{bmatrix} -50 & 80 & 100 \\ 70 & 10 & 0 \\ 530 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 8 \\ 1 \end{bmatrix} = \begin{bmatrix} 690 \\ 150 \\ 530 \end{bmatrix}$$

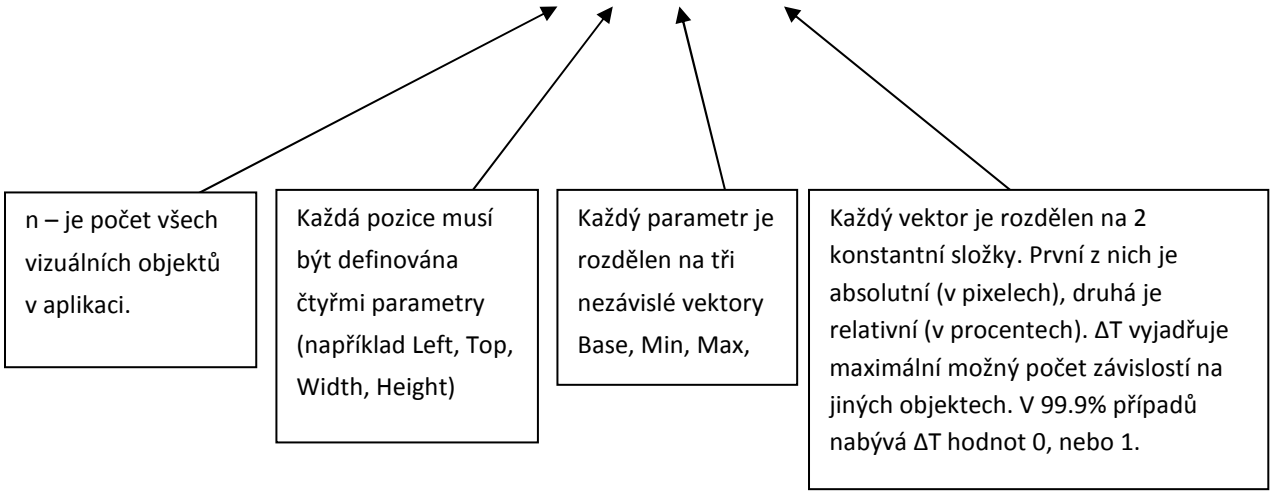
Výsledný vektor udává čísla $\text{Left}_{\text{Base}}$, Left_{Min}, Left_{Max} v pixelech. Nyní již stačí jen vložit tyto čísla do minmax funkce a získat konečný výsledek pro Left.

$$\text{Left} = \text{Min}(\text{Max}(\text{Left}_{\text{Base}}, \text{Left}_{\text{Min}}), \text{Left}_{\text{Max}}) = \text{Min}(\text{Max}(690{,}150), 530) = 530$$

Hodnota Left nakonec vyšla 530 pixelů.

Z výše uvedeného postupu je zřejmé, že pro výpočet pozice libovolného objektu, který je zadán čtyřmi hodnotami, je zapotřebí 4 násobení matice x vektor plus 4 krát rozhodování minmax funkce. Výpočetně nejnáročnější je násobit matici vektorem. Naštěstí tato operace může být hardwarově urychlena.

$$\text{T}(n) = \text{O}(\text{n} * 4 * 3 * (2 + \Delta\text{T}))$$



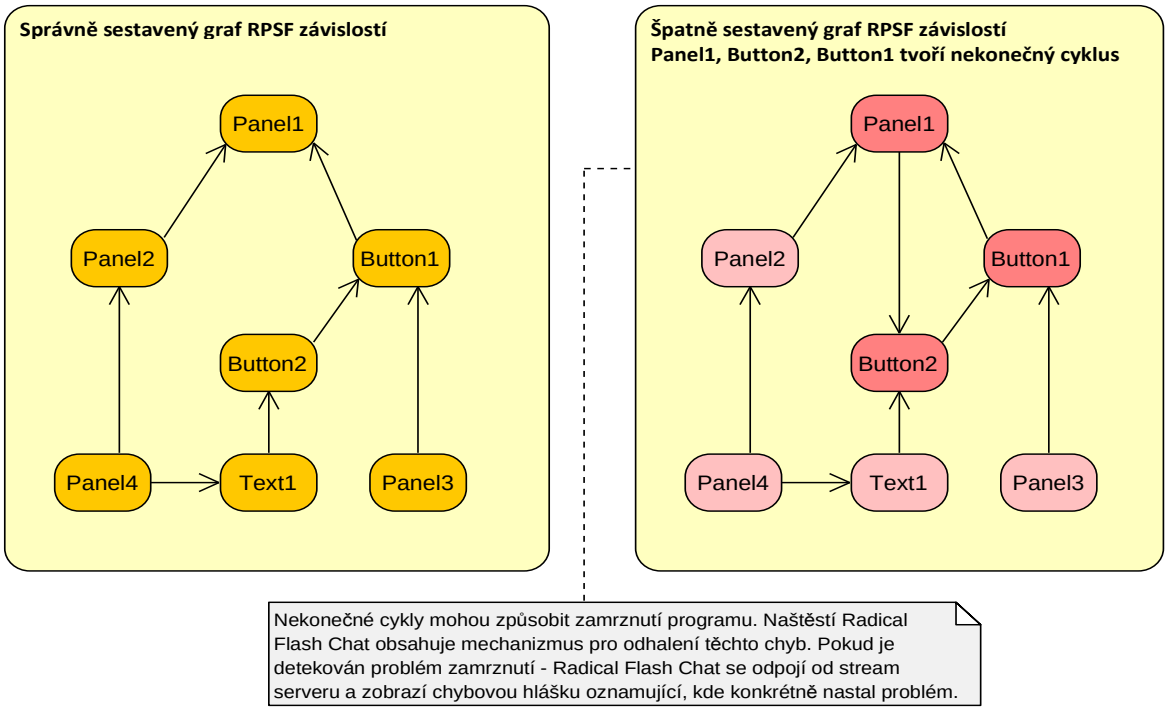
Celý algoritmus výpočtu se tak stává mnohem efektivnější a výkonnější, než je tomu v jazyce HTML doplněném o CSS. Relativní poziční systém uvedený poprvé ve frameworku Radical Chat spojuje relativní i absolutní pozicování známé z kaskádových stylů, umožňuje definovat přetékání pomocí min, max omezení a navíc dovoluje návrhářům provázat pozice jednotlivých objektů mezi sebou. Výsledné navrhované řešení je tedy mnohem jednodušší a efektivnější, než je tomu v jakékoliv jiné běžně používané technologii současnosti.

Pozn. relativní poziční systém je možné otestovat v aplikacích na přiloženém DVD.

4.6.2 Zásady pro použití relativního pozičního systému

Relativní poziční systém je velice efektivní a jednoduchý na použití, nicméně netoleruje žádné chyby a to jak syntaktické, tak logické. Návrháři jej musí používat přesně a korektně. V opačném případě se aplikace vůbec nespustí a zobrazí se chybová hláška. Pravidla pro psaní RPS výrazů jsou uvedena níže:

i. Ve výrazech je dovoleno kombinovat relativní absolutní zložky jako jsou procenta, pixely nebo palce, alo pouze za pomocí znamének plus a mínus. Výsledek výrazu je vždy vypočten v pixelech.
ii. Každý výraz může obsahovat omezení min, nebo max. Hodnota min musí být vždy menší , nebo rovna hodnotě max.
iii. Každý objekt může obsahovat reference na pozice jiných objektů, avšak výsledný graf závislostí musí být vždy orientovaným grafem beze smyček. Pokud tomu tak není aplikace se nespustí a je zobrazena chybová hláška.



Obrázek 13 - graf závislostí

4.7 Univerzální vizuální komponenty

Standardní výbava frameworku disponuje 8 základními vizuálními komponentami, z toho 5 je univerzálních. Jedná se o “BasicPanel”, “HtmlTextArea”, “BasicButton”, “CameraPreview” a “VideoPlayer”. Univerzální komponenty se mohou vyskytovat v libovolném počtu ve vrstvách <BG> a <FG>. Cílem této části textu není důkladně popsat všechny vlastnosti jednotlivých komponent (takový popis je možné najít v manuálu pro vývojáře), ale pouze poukázat na zajímavosti v řešení některých problémů.

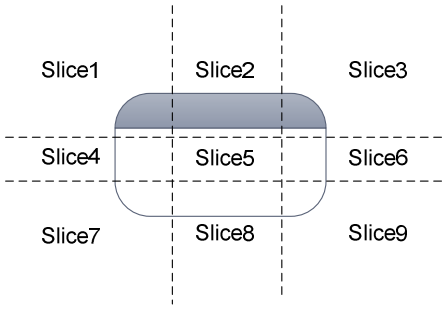
4.7.1 Komponenta BasicPanel

Komponenta BasicPanel není v žádném případě kontejnerem grafických prvků, jak by se mohlo mylně jevit podle názvu. Jedná se o velice sofistikovaný vizuální prvek sloužící především k zobrazování statických obrázků ve formátech PNG, GIF, JPEG. Mezi nejdůležitější vlastnosti patří vlastnost <Pattern> a <NineGrid> sekce <LookAndFeel>.

```
<SkinObject>
  <Type>BasicPanel</Type>
  <Name>Panel</Name>
  <Position>
    <Left>0</Left>
    <Top>0</Top>
    <Right>100%</Right>
    <Bottom>100%</Bottom>
  </Position>
  <LookAndFeel>
    <Pattern>
      <Image>walltexture.jpg</Image>
    </Pattern>
  </LookAndFeel>
</SkinObject>
```

Pattern nastavuje opakovatelnou texturu (vzor), která je použita pro vyplnění oblasti <Position>. Oproti tomu <NineGrid> je určen maticí obrázků dimenze 3x3, jež se používá pro hladké vykreslování zaoblených panelů.

```
<SkinObject>
  <Type>BasicPanel</Type>
  . . .
  <LookAndFeel>
    <NineGrid>
      <Slice1>Slice01.png</Slice1>
      <Slice2>Slice02.png</Slice2>
      <Slice3>Slice03.png</Slice3>
      . . .
      <Slice9>Slice09.png</Slice9>
    </NineGrid>
  </LookAndFeel>
</SkinObject>
```



Obrázek 14 – NineGrid

Ve většině případů není nutné definovat všechny kousky (řezy panelu). Například pokud návrhář grafiky určí jen texturu <Slice5> a ostatní hodnoty slice vynechá, vznikne tím roztahovatelný obrázek, který vyplní oblast <Position>. Aby se návrhářům grafiky pracovalo s jazykem RCSL a komponentou BasicPanel ještě lépe, byl vytvořen zkrácený zápis pomocí něhož mohou nastavit všech devět textur v jednom řádku kódu.

```
<NineGrid>
  <SliceImgSequence>slice{1-9}.png</SliceImgSequence>
</NineGrid>
```

Textový řetězec “{1-9}” nebo “{0-8}” je nahrazen čísly od jedné do devíti případně od nuly do osmi. Samozřejmě u komponenty BasicPanel stejně jako u všech dalších odvozených od SkinObject je možné použít nejrůznější PixelShadery, mixování barev a libovolně měnit alpha kanál.

Komponenta BasicPanel sice patří k nejjednodušším grafickým komponentám multimediálního frameworku Radical Chat, přesto však splňuje veškeré moderní nároky současných web designérů.

4.7.2 Implementace komponenty BasicPanel

Jádro komponenty je odvozeno od třídy SkinObject a dědí všechny jeho vlastnosti. Komponenta je implementována pouze v sekci Radical Flash Chat a jazyku ActionScript. V sekci <LookAndFeel> musí být použita buď podsekce <Pattern>, nebo <NineGrid> takže <LookAndFeel> je pro BasicPanel povinný.

Pro implementaci BasicPanelu je volen poněkud netradiční přístup. Každá komponenta obsahuje na místo jednoho hned dva dynamické virtuální objekty s kapacitou 9 obrázků o rozlišení až 2048x2048 pixelů. Proč právě dva? BasicPanel, jako jednu z mála komponent není možné vytvořit ihned po přečtení RCSL příkazu. K tomu, aby se tak stalo musí být nejprve načten obrázek, nebo

sada obrázků, která určuje vzhled BasicPanelu. Načítání bohužel nějakou dobu trvá – řádově desítky až stovky milisekund. Proto byl zvolen postup asynchronního načítání, který umožní zpracovávat další RCSL příkazy a pokračovat v běhu Radical Flash Chat programu, zatím co se paralelně nahrávají obrázky daného BasicPanelu. Jakmile jsou všechny obrázky nahrány - panel se zobrazí. Zde bohužel může dojít k drobnému zauzlení. Pokud se ve statickém konfiguračním souboru, nebo spíše během dynamického skinování vyskytnou dva RCSL příkazy manipulující vzhledem téhož BasicPanelu mohl by nastat vážný problém.

```
<!-- První deklarace Panelu 1. Je vytvořen BasicPanel s názvem Panel1 -->
<SkinObject>
  <Type>BasicPanel</Type>
  <Name>Panel1</Name>
  . . .
  <LookAndFeel>
    <NineGrid>
      <SliceImgSequence>Slice{1-9}.png</SliceImgSequence>
    </NineGrid>
  </LookAndFeel>
</SkinObject>

<!-- Druhá deklarace Panelu 1. Panel1 je pouze updatován. -->
<SkinObject>
  <Type>BasicPanel</Type>
  <Name>Panel1</Name>
  . . .
  <LookAndFeel>
    <NineGrid>
      <SliceImgSequence>OtherSlice{1-9}.png</SliceImgSequence>
    </NineGrid>
  </LookAndFeel>
</SkinObject>
```

Načítání externích souborů je v Adobe Flashi řešeno asynchronně. Bohužel v takovém případě musí být ve Flashi z jistých důvodů stornování tohoto procesu řešeno rovněž asynchronně. Není proto možné okamžitě zrušit probíhající stahování obrázků a ihned zahájit stahování nové. Pochopitelně takovou situaci by bylo možné řešit převodem na problém synchronní s použitím aktivního čekání. To by značně snížilo celou efektivitu zpracování kódu a neúměrně zpomalilo dynamické skinování (o kterém bude řeč později). Proto byla zvolena zlatá střední cesta. Řečeno z nadhledu BasicPanel v sobě skrývá prostředky pro paralelní načtení dvou virtuálních panelů najednou. Například výše uvedený XML kód je rozpleten následujícím způsobem: Nejprve je vytvořen první virtuální panel pro načtení prvních devíti obrázků Slice{1-9}.jpg v zápětí je načítání asynchronně zastaveno a ihned je vytvořen druhý virtuální panel(loader) pro načtení dalších devíti obrázků tentokrát s názvem OtherSlice{1-9}.jpg. Pak kód pokračuje v běhu dál (vytváří jiné objekty, videoplayery,...). Nějakou dobu trvá načítání prvních devíti obrázků, jejich stornování a paralelně s tím probíhá načítání druhých devíti obrázků. Jakmile je druhé načítání hotovo, je panel

zobrazen. Tento scénář by bylo teoreticky možné rozšířit i pro případ tří, čtyř a více updatů téhož panelu v krátké době za sebou. Nicméně takový postup by zabíral příliš mnoho paměti a systémových prostředků, a proto v momentě kdy nastanou více než dva updaty BasicPanelu v krátké době za sebou se asynchronní paralelní problém synchronizuje a serializuje za cenu delšího zpracovávání RCSL příkazů. Je nutné podotknout, že v praxi se více než dva dynamické updaty jednoho BasicPanelu takřka vůbec nevyskytují a pokud ano, pak jde vždy o chybu vývojáře (používajícího framework Radical Chat nesprávným způsobem), nikoliv o nedostatek Radical Chatu a dynamického skinování.

4.7.3 Komponenta HtmlTextArea

HtmlTextArea slouží k zobrazování textových popisků formátovaných pomocí HTML tagů. Jelikož je celý popisný dokument a jazyk RCSL postaven na formátu XML, vzniká zde jeden nepřehlédnutelný problém. Jak umožnit návrhářům vzhledu aplikací vkládat HTML tagy do těla XML tagů? Řešení jsou hned tři. Vývojář může použít XML entity < a >,

```
<HtmlText>
  &lt;FONT SIZE="12" FACE="_sans" COLOR="#FFAA00" LETTERSPACING="0"&gt;
     SOME TEXT
  &lt;/FONT&gt;
</HtmlText>
```

nebo pomocného formátování CDATA, které vypadá rovněž neúhledně, a nebo dvouzávorkovacího systému (viz. kód níže), jenž se nejvíce osvědčil a je pro komponentu RCSL plně implementován.

```
<SkinObject>
  <Type>HtmlTextArea</Type>
  <Name>HtmlText1</Name>
  <Position>
    <Left>15</Left>
    <Top>100</Top>
    <Width>400</Width>
    <Height>100</Height>
  </Position>
  <HtmlText>
  {{FONT SIZE="12" FACE="_sans" COLOR="#FFAA00" LETTERSPACING="0"}}
     SOME TEXT
  {{/FONT}}
  </HtmlText>
  <IsTextSelectable>false</IsTextSelectable>
</SkinObject>
```

4.7.4 Implementace komponenty HtmlTextArea

Implementace komponenty HtmlTextArea není nikterak složitá a vychází přímo z vizuálního ovládacího prvku TextArea, který je obsazen v jazyce ActionScript 3.0. Jediný problém, který implementace tohoto vizuálního prvku řeší je přeformátování "HtmlTextu" a jeho dvouzávorkovacího systému na HTML tagy a jednoduchý test, který odhalí hrubé nesrovnalosti v HTML formátování.

4.7.5 Komponenta BasicButton

BasicButton je tlačítko, které může sloužit k zobrazování nejrůznějších dialogů, jež přímo ovlivňují chování uživatelské aplikace, ale také ke komunikaci se serverem. Nejdůležitější vlastností tohoto vizuálního ovládacího prvku je událost <Action> parametru <OnClick>.

```
<OnClick>
  <Action>onUniversalCall_WithSelectedClient</Action>
</OnClick>
```

Action přesně určuje, co se má stát jakmile uživatel klikne na tlačítko. Mohou nastat tři situace:

1. Po stisknutí tlačítka je zobrazen lokální modální dialog. Například pro výběr barvy textu, výběr audio a video vstupu (chce-li klient začít publikovat stream), atd… Tato událost není hlášena serverové části aplikace.
2. Může nastat lokální akce, která se následně promítne do celého systému (zobrazí všem připojeným klientům). Touto akcí je například odeslání textové zprávy – tedy textu obsaženém v komponentě ChatInput, podobně jako by uživatel po dopsání textové zprávy stisknul klávesu enter.
3. Může nastat globálně neurčitá akce. Klient vyvolá v serverové části aplikace určitou událost jejíž zpracování se může dostat až do vrstvy Radical Web Service.

Použití konkrétní události dále určí jak má vypadat forma zápisu BasicButtonu V RCSL. Samotný BasicButton je velice složitá komponenta, která specifikací konkrétní akce nabývá velké množiny rozličných vlastností, jež může vývojář změnit.

Akce implementované v současné verzi Radical Chatu jsou:

1. ShowDialog_SelectColor, ShowDialog_SelectEmoticon.
2. SendTextMessage, ShowDialog_SelectCamera.
3. onUniversalCall, onUniversalCall_WithSelectedClients.

Má-li se stisknutím tlačítka vyvolat dialog pro výběr barvy textu, pak vypadá zápis následovně:

```
<OnClick>
  <Action>ShowDialog_SelectColor</Action>
  <!-- Paleta barev může nabývat až 20 hodnot. -->
  <Palette>
    <Color>990000</Color>
    <Color>FF0000</Color>
    <Color>CC9933</Color>
    <Color>FA9100</Color>
    . . .
  </Palette>
</OnClick>
```

Jak je vidět z příkladu výše - uvedení akce ShowDialog_SelectColor otevírá cestu ke skryté vlastnosti <Palette>, kde může být specifikováno až 20 základních barev palety daného modálního dialogu.

Další zajímavou akcí je ShowDialog_SelectCamera.

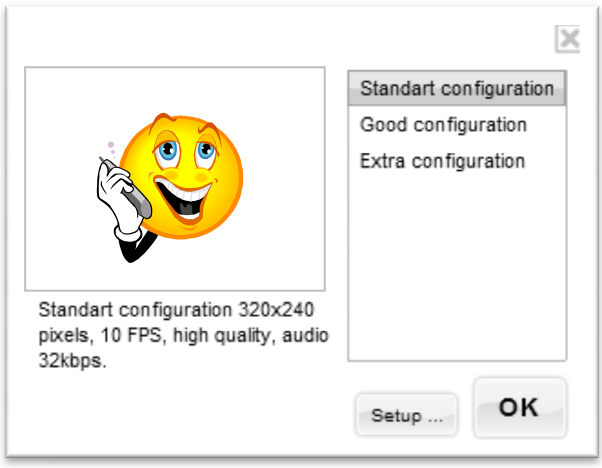
```
<OnClick>
  <Action>ShowDialog_SelectCamera</Action>
  <!--
    Dialog může být zobrazen ihned poté co je tlačítko vytvořeno.
    Toho se využívá především v dynamickém skinování.
  -->
  <AutoShowDialog>false</AutoShowDialog>
  <!-- Konfigurace pro streamování. -->
  <StreamingConfiguration>
    <!-- Název streamu -->
    <StreamName>Stream1</StreamName>
    <!--
      Seznam možností ze kterých si uživatel může v dialogu vybírat.
    -->
    <AVOption>
      <!-- Krátká textová reprezentace názvu streamovací možnosti. -->
      <Title>Standart configuration</Title>
      <!-- Dlouhý textový popis zobrazený v dialogu. -->
      <Description>
        Standart configuration 320x240 pixels, 10 FPS, high quality.
      </Description>
      <!-- Je tato možnost streamování v seznamu vybrána jako výchozí -->
      <IsDefault>true</IsDefault>
      <!-- Nastavení zvuku -->
      <Audio>
        <!-- Audio kilobit rate -->
        <Kbitrate>32</Kbitrate>
        <!-- Potlační šumu. Hodnota z intervalu <0;100> -->
        <SilenceLevel>5</SilenceLevel>
      </Audio>
      <!-- Nastavení videa -->
      <Video>
        <!-- Rozlišení kamery v němž bude stream zaznamenáván. -->
```

```
        <Resolution>
          <Width>320</Width>
          <Height>240</Height>
        </Resolution>
        <!-- Video kilobit rate -->
        <Kbitrate>320</Kbitrate>
        <!-- Snímků za vteřinu -->
        <FPS>10</FPS>
        <!-- Kvalita videa. Minimální hodnota 10, maximální 100. -->
        <Quality>80</Quality>
        <!-- Interval - jak často má být zaznamenáván klíčový snímek -->
        <KeyFrameInterval>30</KeyFrameInterval>
      </Video>
    </AVOption>

    <AVOption>
    ...
    </AVOption>

    <AVOption>
    ...
    </AVOption>
  </StreamingConfiguration>
</OnClick>
```

S pomocí vlastnosti <StreamingConfiguration> má návrhář grafiky možnost určit v jaké kvalitě a rozlišení bude moci uživatel publikovat audiovizuální multimediální obsah. Jedná se o dialog druhé kategorie přímo pracující se serverovou částí aplikace.



Obrázek 15 - dialog SelectCamera

Jakmile je na počítači uživatele povolena práce s kamerou a vybrána některá z přednastavených možností <AVOption> sekce <StreamingConfiguration> je na stream serveru téměř instantně vyvolána událost onClientStartedStreaming a předána ke zpracování webové službě RWS (Viz. kapitola 3). Webová služba by měla z bezpečnostních důvodů vždy znovu ověřit zda-li je uživatel

autorizován publikovat multimediální obsah. Živý stream je poté přijímán a nahráván stream serverem. Pokud se má nahrávaný soubor dat (většinou video) zobrazit i ostatním uživatelům, musí to vývojář zabezpečit kooperací s komponentou VideoPlayer.

Posledním typem událostí je onUniversalCall a onUniversalCall_WithSelectedClients. Tyto akce nezobrazují žádný konkrétní dialog a nevyžadují bližší interakci s uživatelem. Informace o stištění tlačítka ihned probublá celým systémem až do vrstvy Radical Web Service, kde spustí kód funkce onUniversalCall. V těle funkce může vývojář libovolně manipulovat vnitřním chováním Radical Bridge, nebo koncových uživatelských aplikací.

4.7.6 Implementace komponenty BasicButton

Implementace této komponenty patří k jedné z nejsložitějších vůbec. Jednak je to zapříčiněno velkým množstvím různorodých akcí, ale také ošetřováním podivuhodných kritických stavů.

Jádro komponenty je odvozeno od třídy SkinObject a dědí všechny jeho vlastnosti. V sekci <LookAndFeel> byla vytvořena povinná podsekce <Images> s vlastnostmi <NormalImage>, <OverImage>, <DownImage>, takže celá sekce <LookAndFeel> je pro BasicButton povinná podobně jako tomu je u BasicPanelu. Ke zvláštnímu druhu ochrany vyvinutému speciálně pro Radical Chat patří časové zámky tlačítek. Po uvolnění první verze Radical Chatu 0.8.5 byla několikrát identifikována situace při níž došlo k zahlcení slabších serverů přebytečnými událostmi onClick. Proto byl vyvinut mechanizmus časových zámků. Pokaždé, když uživatel stiskne nějaké tlačítko aplikace a vyvolá akci onClick zamrznou pro daného uživatele všechna tlačítka na dobu časového intervalu t. V poslední verzi systému je interval t nastaven na půl vteřiny. Další vlastností, která stojí za zmínku je <AutoShowDialog> pro akce první kategorie. Pokud je tato vlastnost nastavena na hodnotu “true”, pak se dialog zobrazí ihned po vytvoření komponenty BasicButton podobně jako by uživatel sám dané tlačítko stisknul. V tomto případě je ale záměrně ignorován mechanizmus časových zámků. K čemu je takováto vlastnost dobrá? Slouží především pro použití spolu s dynamickým skinováním (o kterém bude řeč později) a dává vývojářům možnost vytvořit neviditelné tlačítko a ihned na něm spustit metodu onClick - což zabezpečí okamžité zobrazení dialogu. Poslední zvláštností je akce ShowDialog_SelectCamera a sekce StreamingConfiguration, ve které může návrhář grafiky nastavit rozlišení a kvalitu nahrávaného audia a videa. Něco takového dosud nebylo možné. Většina stream serverů má kvalitu obrazu a zvuku nastavenu v externím konfiguračním souboru, který není přístupný ostatním uživatelům. Ani samotný Flash neobsahuje komponentu, která by přímo umožňovala nastavit parametry publikovaného streamu – vše se musí dělat poměrně těžkopádně přes jazyk ActionScript. Radical Chat a ovládací prvek BasicButton přinášejí poprvé tuto možnost i běžným návrhářům grafiky, kteří takto mohou testovat a lépe sladit vzhled aplikace a audiovizuální obsah, který v ní bude přehráván (případně pomocí ní bude streamován směrem vzhůru na server). Velikou otázkou

která vyvstává z takovéhoto přístupu je bezpečnost. Samozřejmě i méně schopný hacker, či schopnější uživatel dokáže relativně jednoduše odchytit síťové packety obsahující konfigurační soubor, který přichází do Radical Flash Chatu v době jeho spuštění. Podvrhnout v něm parametr <StreamingConfiguration> není nic těžkého. Naštěstí i pro tento případ má Radical Chat (Radical Bridge) záchytný mechanizmus. Jestliže se klient pokusí streamovat audio a video v jiném než povoleném rozlišení a kvalitě, pak ještě před tím, než je takový stream přijat musí jej schválit funkce onClientWantsToStartStreaming, kde vývojář může otestovat skutečné parametry publikovaného videa a případně streamování zakázat.

4.7.7 Komponenta VideoPlayer

Vizuální grafický prvek VideoPlayer bývá zpravidla vytvářen pouze dynamicky, téměř nikdy ne staticky. Jak již sám název napovídá VideoPlayer slouží pro přehrávání videa nebo též audia. Videem může být již sestříhaný a předpřipravený film umístěný na serveru, nebo živý obraz publikovaný některým z klientů. Deklarace videoplayeru není nikterak komplikovaná

```
<SkinObject>
  <Type>VideoPlayer</Type>
  <Name>PerformerVideo</Name>
  <StreamUID>roomUID/clientUID/streamName</StreamUID>
  <Visible>true</Visible>
  <Smooth>true</Smooth>
  <Position>
    <Left>15</Left>
    <Top>55</Top>
    <Width>388</Width>
    <Height>290</Height>
  </Position>
</SkinObject>
```

avšak při podrobnějším pohledu se dá nalézt jeden menší problém. Hodnota vlastnosti <StreamUID> nemusí být vždy zcela jednoznačná. Pokud je přehrávaným multimediálním obsahem myšleno video umístěné na serveru, pak <StreamUID> určuje cestu k tomuto videu. Má-li se však přehrávat živý stream, pak <StreamUID> určuje globálně jednoznačný identifikátor živého proudu dat, který v daný okamžik nemusí nutně existovat, nebo nemusí být validní. <StreamUID> pro živé video je vytvořen teprve poté, co konkrétní uživatel začne publikovat stream. Proto je ve většině případů VideoPlayer vytvářen dynamicky v rámci události onClientWantsToStartStreaming a téměř nikdy ne staticky.

4.7.8 Implementace komponenty VideoPlayer

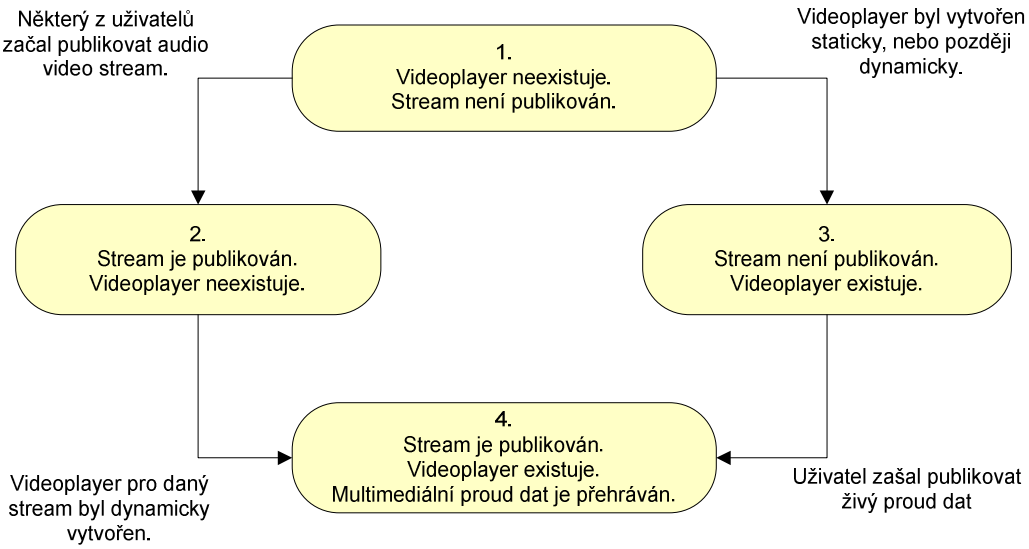
Komponenta VideoPlayer patří k jedněm z mála vizuálních prvků, kde je použití komponenty (návrháři grafiky a uživateli frameworku) výrazně jednodušší, než její vnitřní implementace. VideoPlayer může běžet ve třech módech:

1. On-demand video - statické video stahované ze serveru, v budoucnosti může být řešeno přímo jazykem HTML5 a tagem video.
2. Live-room streaming. Živý multimediální obsah publikovaný klientem dané chatové místnosti. Audio a video je možné publikovat nejen prostřednictvím Adobe Flash Playeru, ale také pomocí freewarového nástroje Flash Media Live Encoder, či VLC. V takovém případě je možné použít video kodeků On2 VP6 a H.264.
3. Live-cross-room streaming. Živý multimediální obsah publikovaný klientem jiné chatové místnosti, než ve které se nachází uživatel, který chce video přehrávat.

VideoPlayer především při zobrazování živého multimediálního proudu dat (scénář 2 a 3) musí řešit několik ne příliš zřejmých problémů. Teoreticky mohou pro každý přehrávač nastat čtyři elementární situace, kterými prochází aplikace od svého spuštění do ukončení.

Z pohledu klientské aplikace, která byla právě spuštěna se může stát:

1. V místnosti není publikován konkrétní živý stream a ani aplikace pro něj nemá připravený žádný přehrávač.
2. V rámci místnosti je publikován stream, ale aplikace pro něj nemá nastavený přehrávač.
3. Aplikace má připravený přehrávač, ale živý stream není v rámci místnosti publikován.
4. V místnosti je publikován stream a aplikace jej přehrává v korektně nastaveném přehrávači s patřičným <StreamUID>.



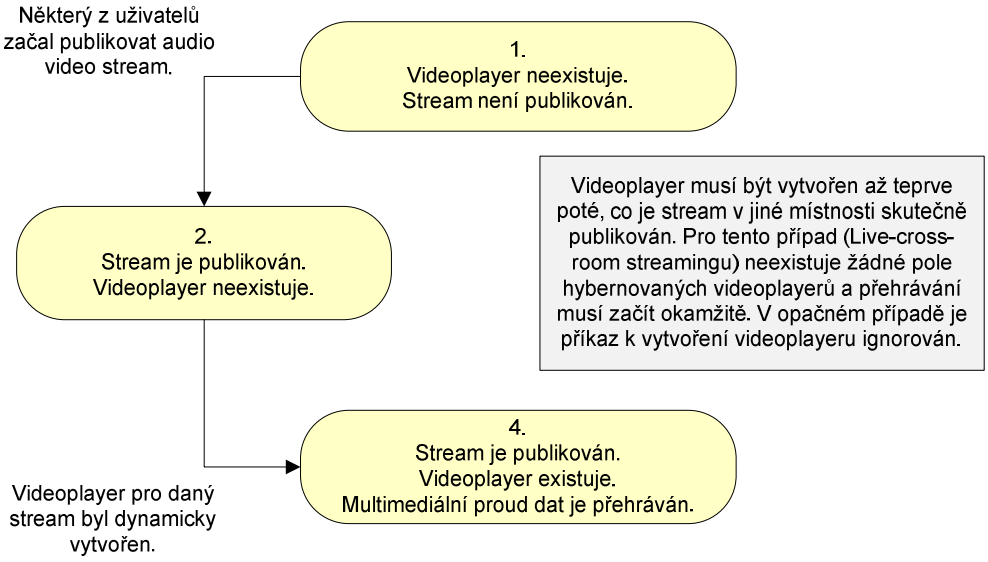
Obrázek 16 - vnitřní stavy videoplayeru

Jednotlivé stavy se mohou vzájemně měnit podle obrázku výše. Stav 1 je výchozí. Do stavu 2 se může aplikace dostat změnou dynamického kontextu - během načítání seznamu uživatelů. Do

stavu 3 může být aplikace uvedena jak ze statického kontextu, tak pomocí dynamického skinování. Navíc přehrávačů a publikovaných streamů může být až nekonečně mnoho. Proto bylo pro zbylé scénáře po delších úvahách zvoleno následující řešení.

Radical Flash Chat monitoruje stav všech klientů v dané místnosti, ke které je připojena aplikace koncového uživatele. V případě, že některý z nich zahájil sdílení živého audia či videa je sekvenčně prohledáno pole "spících" videoplayerů. Jakmile v něm Radical Flash Chat narazí na hledaný <StreamUID>, pak vytvoří (probudí) daný VideoPlayer a zahájí přehrávání živého proudu dat. Tím je řešen přechod ze stavu 3 do stavu 4. Obdobně pro přechod ze stavu 2 do stavu 4: pokud je náhle dynamicky změněn skin aplikace a má se vytvořit nový videoplayer, pak ještě před tím než je přidán do struktury spících přehrávačů je prohledáno pole publikovaných živých streamů dané místnosti a případně hned zahájeno přehrávání.

Pokud jde o situaci Live-cross-room streaming vše je velice podobné až na to, že byl vynechán stav číslo 3. Klienti v jedné chatovací místnosti jsou vždy připojeni k jednomu fyzickému serveru. Nicméně různé místnosti mohou být provozovány na různých serverech propojených vnitřní sítí nebo pomocí VPN. V praxi se tak sice neděje, to ale neznamená, že to není možné. Jelikož by sledování všech (až tisíců) klientů ve všech místnostech na všech serverech systému Radical Chat bylo takřka nemožné a z technických důvodů neproveditelné - tvůrce frameworku se rozhodl, že větev procházející stavem 3 nebude v Live-cross-room streamingu podporována.



Obrázek 17 - cross-room streaming

Ovládací prvek VideoPlayer je mocnou zbraní především ve spojení s dynamickým skinováním. Uživatelé frameworku Radical Chat se nemusí starat o problémy spojené s navazováním spojení,

neustálou kontrolou stability proudu dat či šířky pásma jednotlivých klientů. O vše se stará komplikovaná komplexní implementace komponenty VideoPlayer.

4.7.9 Komponenta CameraPreview

Grafický prvek CameraPreview slouží k zobrazení záběru ze záznamového zařízení (většinou webové kamery) ještě před tím než se záběr zkomprimuje a odešle na server formou živého proudu dat. Tato komponenta je úzce spjata s tlačítkem BasicButton a akcí ShowDialog_SelectCamera. Použití komponenty je velice jednoduché.

```
<SkinObject>
  <Type>CameraPreview</Type>
  <Name>MyCamera</Name>
  <StreamName>Stream1</StreamName>
  <Visible>true</Visible>
  <Position>
    <Left>155</Left>
    <Top>100%-75</Top>
    <Width>80</Width>
    <Height>61</Height>
  </Position>
</SkinObject>
```

Vlastnost <StreamName> je povinná a její hodnota by se měla shodovat s hodnotou <StreamName> některého tlačítka BasicButton, pokud v aplikaci existuje alespoň jedno takové tlačítko sloužící pro vyvolání dialogu “Select Camera”.

4.7.10 Implementace komponenty CameraPreview

Implementace vizuálního grafického prvku CameraPreview není zdaleka tak složitá, jako je tomu u dříve zmiňovaného VideoPlayeru. Použití komponenty nevyžaduje žádnou spolupráci se serverovou částí aplikace a nezatěžuje tak datový tok ostatních uživatelů. Podobně jako u VideoPlayeru se i v tomto případě vyskytuje v aplikaci klienta něco jako “spící” pole objektů typu CameraPreview. Ty se aktivují podle potřeby v reakci na dialog “Select Camera” vyvolaný stisknutím tlačítka BasicButton.

4.8 Specializované komponenty

Specializované komponenty, občas také nazývané hanlivým přízviskem toxické se v Radical Chatu vyskytují od první uvolněné verze 0.8.5 až po aktuální verzi 1.0.0.1455. Mohou být v rámci celé uživatelské aplikace použity maximálně jednou a to jen ve vrstvě <ML>. Jejich primárním určením je zjednodušení vývoje služeb pro online textové chatování. I když se tvůrce multimediálního frameworku několikrát pokoušel nalézt cestu, jak dané komponenty převést na univerzální, prozatím se mu to nepodařilo.

Balík specializovaných komponent aktuálně čítá tři vizuální grafické prvky ne nepodobné HtmlTextArea. Jmenovitě se jedná o ChatInput, ChatText a UsersList.

Komponenta UsersList slouží k zobrazení netřízeného seznamu všech uživatelů, kteří jsou připojeni k dané místnosti.

ChatInput je jednoduché vstupní políčko do kterého může uživatel vkládat text a následně jej odeslat klientům vybraným v seznamu UsersList. Samozřejmě textová zpráva ještě před tím putuje na stream server, případně na web server a může být upravena v proceduře onClientSendTextMessage.

ChatText slouží k zobrazení textových zpráv určených přímo pro daného klienta, nebo chatovací místnost. Na rozdíl od předešlých dvou komponent obsahuje i poněkud komplikovanější vlastnosti, a proto je jí věnován krátký rozšiřující odstavec.

4.8.1 Komponenta ChatText

ChatText zpracovává textovou zprávu a převádí jí z formátu PTM (plain text message) do zobrazitelné, uživatelsky přívětivé podoby. Hlavní výhodou je zde fakt, že převod se uskutečňuje až na počítači, nebo obecně libovolném koncovém zařízení klienta. Tím pádem je možné zajistit výborné propojení funkce textového dopisování standardních PC s chytrými kapesními zařízeními, nebo mobilními telefony. Navíc při příchodu nové textové zprávy je možné vyvolat některou z předprogramovaných událostí - nechat zahrát hudbu, změnit skin, atp. To dává vývojářům široké možnosti seberealizace a šetří čas jinak nutný k vývoji běžného textového chatu. Hlavní výhodou komponenty ChatText je šablonovací systém. Ten se stará o převod čisté textové zprávy do HTML kódu. Díky tomu se zpráva ve formátu PTM může přizpůsobit různým zařízením, jejich rozlišení a formátování.

Následující příklad použití komponenty ChatText je uveden za použití dvouzávorkové notace:

```
<SkinObject>
  <Type>ChatText</Type>
  <Name>ChatText</Name>
  <Visible>true</Visible>
  <Position>
    <Left>430</Left>
    <Top>45</Top>
    <Right>100%-10</Right>
    <Bottom>100%-15</Bottom>
  </Position>
  <!-- Zapíná, nebo vypíná zobrazování emoticons(smajlíků) -->
  <EmoticonsEnabled>false</EmoticonsEnabled>
  <!-- Výchozí text zobrazený v těle komponenty -->
  <HtmlText>
    {{FONT SIZE="12" COLOR="#FFAA00"}}Welcome in the room.{{/FONT}}
  </HtmlText>
  <!-- Zvuk přehrávaný pokaždé,když dorazí nová textová zpráva -->
  <NewMessageSound>./skin/sounds/newmessage.mp3</NewMessageSound>
  <!--
    Šablona pro převod veřejné textové zprávy z PTM do HTML formátování
    Je možné použít následující výrazy vztahující se k PTM
    {message.getColor()} - vrátí hexadecimální kód PTM textové zprávy,
    {message.sender.getNickName()} - uživatelské jméno odesilatele zprávy
    {message.getText()} - text PTM zprávy
    {datetime.format("string")} - zobrazí datum a čas přijetí zprávy.
                                  Je nutné uvést formátovací řetězec.
      %a - anglické značení času am, pm,    %A - anglicky AM nebo PM
      %d - den v měsíci 01-31,              %D - den v měsíci 1-31
      %g - 12-ti hodinová notace 00-11,     %G - hodiny 0-11
      %h - 24 hodinová notace 00-23,        %H - hodiny 0-23
      %i - minuty 00-59,                    %I - minuty 0-59
      %m - číslo měsíce v roce 01-12,       %M - číslo měsíce v roce 1-12
      %N - název měsíce (January),          %n - ID měsíce (Jan)
      %s - vteřiny 00-59,                   %S - vteřiny 0-59
      %y - poslední 2 číslovky roku         %Y - poslední 4 číslovky roku
  -->
  <PublicMessage>
    <Template>
    {{P ALIGN="LEFT"}}
      {{FONT SIZE="12" COLOR="#{message.getColor()}" LETTERSPACING="1" }}
      {{B}}{message.sender.getNickName()}{{/B}}
      {{/FONT}}
    {{/P}}

    {{P ALIGN="LEFT"}}
      {{FONT FACE="_sans" SIZE="12" COLOR="#{message.getColor()}"}}
      {message.getText()}
      {{/FONT}}
    {{/P}}

    {{P ALIGN="RIGHT"}}
      {{FONT FACE="_sans" SIZE="8" COLOR="#000000"}}
      {datetime.format("%Y.%m.%d %h:%i:%s")}
      {{/FONT}}
    {{/P}}
    </Template>
  </PublicMessage>
```

```
  <!--
    Šablona pro převod veřejné textové zprávy z PTM do HTML formátování.
    Lze využít všech možnosti uvedených v popisu sekce PublicMessage.
    Navíc je možno použít:
    {message.receiver.getNickName()} - uživatelské jméno příjemce
    {message.receiver.getUID()} - jedinečný identifikátor příjemce
  -->
  <PrivateMessage>
    <Template>
    {{P ALIGN="LEFT"}}
      {{FONT SIZE="12" COLOR="#{message.getColor()}" LETTERSPACING="1"}}
      {{B}}{{I}}
      {message.sender.getNickName()} &amp;gt;&amp;gt;
      {message.receiver.getNickName()}
      {{/I}}{{/B}}
      {{/FONT}}
    {{/P}}

    {{P ALIGN="LEFT"}}
      {{FONT FACE="_sans" SIZE="12" COLOR="#{message.getColor()}"}}
      {message.getText()}
      {{/FONT}}
    {{/P}}

    {{P ALIGN="RIGHT"}}
      {{FONT FACE="_sans" SIZE="8" COLOR="#000000" }}
      {datetime.format("%Y.%m.%d %h:%i:%s")}
      {{/FONT}}
    {{/P}}
    </Template>
  </PrivateMessage>
  <!--
    Délka historie zpráv.
  -->
  <MessagesBufferSize>15</MessagesBufferSize>
</SkinObject>
```

Mezi nejdůležitější vlastnosti patří <Template> sekce PublicMessage a <Template> sekce PrivateMessage. Rozdíl mezi veřejnými a osobními zprávami je ne přímo patrný, ale dosti podstatný. Veřejné zprávy jsou sdílené takzvanými public shared objekty a danou textovou zprávu vidí všichni uživatelé chatové místnosti. Navíc je na serveru udržována a také logována historie veškeré veřejné komunikace, takže nově příchozí klient si může přečíst o čem se v dané místnosti hovořilo ještě před tím než do ní vstoupil. Osobní zprávy jsou posílány přímo konkrétním osobám. Ve výchozím nastavení nejsou logovány a prakticky ihned po doručení je systém vymaže z paměti. Proto obsahuje komponenta ChatText dvě šablony. Jednu zvlášť pro veřejné, druhou pro osobní zprávy. Implementace této komponenty neskýtá žádná úskalí, proto zde nebude popisována.

5 Dynamický kontext aplikace

Dynamickým kontextem aplikace je myšleno především dynamické skinování a záležitosti s ním spojené. Dynamické skinování je jednou z největších předností a zcela unikátních vlastností Radical Chatu. Umožňuje vývojářům měnit vzhled a chování aplikace za běhu. Vše je řízeno směrem shora dolů. Serverová část ovládá změny jednotlivých grafických uživatelských prostředí, nebo celých skupin uživatelských prostředí. Je nutno podotknout, že žádný z konkurenčních systémů nemá ve svém jádru zabudovánu podobnou funkcionalitu.

5.1 Dynamické skinování

Vzhled a základní chování každé aplikace je určeno statickým kontextem, ještě před tím, než se daná koncová uživatelská aplikace připojí do systému Radical Chat (Viz. kapitola Statický kontext). Podobně jako při programování Windows Forms, Java Swing, nebo C# Forms WPF i v Radical Chatu mají vývojáři možnost měnit vzhled a chování aplikace za běhu, avšak zcela odlišným způsobem, než jak jsou zvyklí z jiných technologií. Rozdíl je v asynchronním multi-uživatelském provedení - změny jsou vykonávány pro celé skupiny uživatelů jediným příkazem, avšak s určitým časovým zpožděním. Jelikož je celý systém vysoce distribuovaný a paralelizovaný, je velice těžké dodržet správné pořadí zpracování příkazů a může docházet k naprosto neočekávaným událostem a stavům. Naštěstí uživatelé frameworku si s tímto nemusí lámat hlavu, neboť veškerá politika dynamického skinování spolu s problémy úplné synchronizace jsou zapouzdřeny v jádře Radical Bridge.

Pro naprostou spokojenost a pohodlí vývojářů řeší problém dynamického skinování pouhopouhé dvě funkce. Jejich použití je velmi snadné a rychlé.

```
function _onClientWantsToStartStreaming(&$client, $stream)
{
  Logger::func("onClientWantsToStartStreaming");
  //client can publish the stream
  $client->setStreamingAccepted();

  //get StreamServerCommand class for communication
  //with Radical Bridge
  $ssc = $client->getStreamServerCommands();
  //try to retrive Skin manipulator for $client
  $skin = $ssc->client_getSkin($client->getUID());
  //update or create SkinObject Panel5 in client's web browser
  $skin->updateSkinObject("
    <Skin>
      <BG>
        <SkinObject>
```

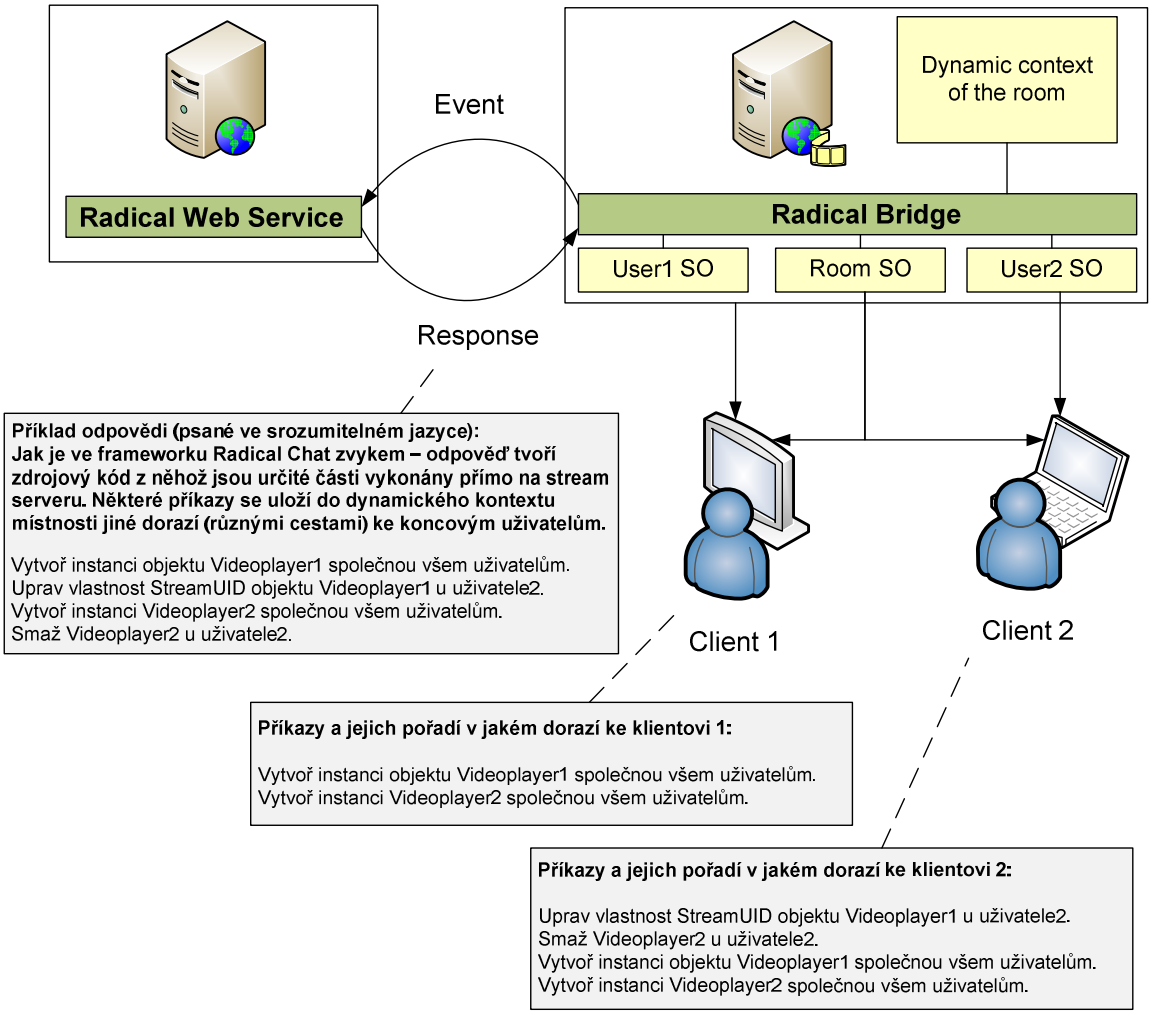
```
          <Type>BasicPanel</Type>
          <Name>Panel5</Name>
          <Visible>true</Visible>
          <Position>
            <Left>50</Left>
            <Top>-40</Top>
            <Right>410</Right>
            <Bottom>360</Bottom>
          </Position>
          <LookAndFeel>
            <NineGrid>
              <SliceImgSequence>
                ./skin/panels/rsp1_0{1-9}.png
              </SliceImgSequence>
            </NineGrid>
          </LookAndFeel>
        </SkinObject>
      </BG>
    </Skin>
  ");

  //get room skin manipulator
  $room_skin = $ssc->room_getSkin($client->getRoomUID());
  //destroy dynamically created object ButtonStartStreaming
  $room_skin->deleteAllSkinObjectUpdates("ButtonStartStreaming");
}
```

V prostředí PHP je možné vyhnout se vkládání XML kódu popisujícího grafiku přímo do těla skriptu použitím funkce updateSkinObjectFromFile (V podstatě dělá totéž co updateSkinObject. Jen XML zdrojový kód je uložen v souboru.).

5.2 Implementace dynamického skinování

Implementace dynamického skinování a řešení přidružených problému tvoří značnou část jádra celého frameworku. I když příklad uvedený v kapitole 5.1 na první pohled nevypovídá o přílišné složitosti a komplexnosti problému opak je pravdou. Ucelenější pohled do této problematiky poskytuje obrázek níže.



Obrázek 18 - dynamické skinování

Z obrázku je patrné, jakým způsobem jsou příkazy operující se změnami chování a vzhledu aplikací přenášeny celým systémem. Z důvodu bezpečnosti, na níž je kladen veliký důraz, jsou tzv. privátní a veřejné změny (alterations) přenášeny a synchronizovány různými síťovými kanály a mohou tedy k cíli dorazit v rozdílném čase a pořadí (Viz. obrázek výše - pořadí příkazů u Klienta 2). Navíc se do systému mohou náhodně připojovat další klienti a staří již nalogovaní jej nenadále opouštět. Aby

toho nebylo málo může v jeden okamžik nastat až N rozdílných paralelních událostí, které do doposud komplikované hierarchie příkazů vnesou naprostý chaos. K následné synchronizaci je nutné znát nejen sekvenční ID (časovou značku a revizní číslo jednotlivých objektů), ale také kontext (událost) z níž příkaz vzešel. Kromě toho existují čtyři závazná pravidla (podmínky), které musí vývojáři používající dynamické skinování striktně respektovat. Tato pravidla pomáhají v kritických situacích rozplést řetězec událostí a nastolit správné pořadí příkazů produkovaných částí Radical Web Service.

1. Celistvost - Funkce pro úpravu vzhledu updateSkinObject opravuje chování právě jednoho objektu, nebo grafické komponenty v aplikaci klienta, či skupiny několika klientů. Vyžaduje-li vývojář změnu chování více objektů najednou - musí použít funkci updateSkinObject vícekrát.
2. Jednoznačnost - XML kód předaný jako parametr updateSkinObject musí jednoznačně specifikovat kompletní cestu ke komponentě (včetně tagu <Skin> a názvu vrstvy, kde daná komponenta leží). Situace se dá přirovnat k absolutní cestě v souborovém systému. Radical Chat relativní cesty nepodporuje.
3. Úplnost - XML kód musí být vždy úplný - tzn. obsahuje i parametry jejichž hodnota nebyla od posledního zásahu změněna. Tento požadavek je poněkud neobvyklý, avšak má své opodstatnění. Každá nová instance grafické, či nevizuální komponenty se po svém vytvoření nachází ve výchozím stavu. Její vlastnosti a události mají tzv. defaultní nastavení. Z mnoha vývojových prostředí a programovacích jazyků jsou vývojáři zvyklí postupovat krok po kroku a sekvenčně měnit hodnoty vlastností dané komponenty.

   ```
   Form1.Button1.Width = 100; Form1.Button1.Height = 100;
   Form1.Button1.Text = “Send”; Form1.Button1.Enabled = true;
   //Později jinde v kódu
   Form1.Button1.Enabled = false;
   ```

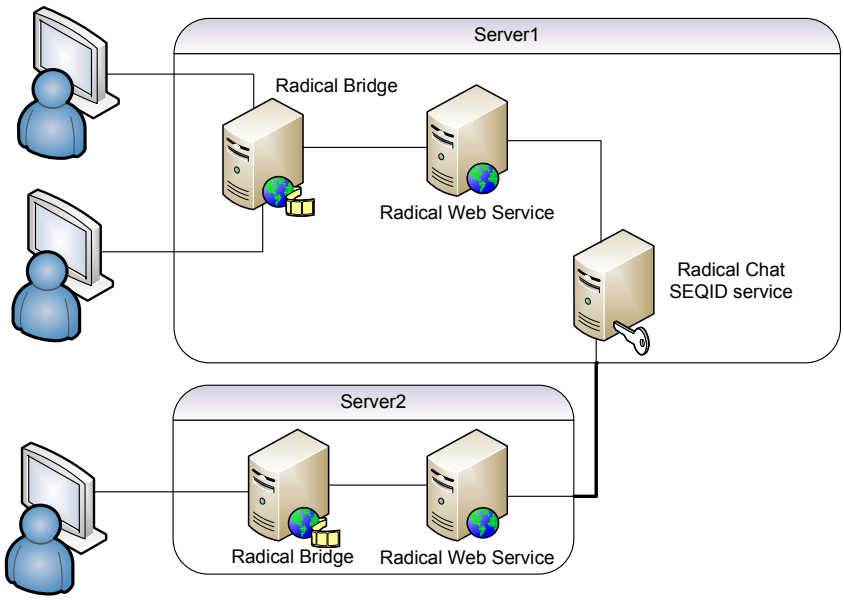
   Chce-li programátor změnit chování komponenty v Radical Chatu, musí uvést všechny měněné parametry v jednom příkazu. Potřebuje-li však později opravit hodnotu byť jedné jediné vlastnosti téže instance komponenty, musí uvést všechny parametry lišící se od defaultního nastavení znovu.

   ```
   <Skin><BG>Button1{Width:100;Height:100;Text:Send;Enabled:true;}</BG></Skin>
   //Později jinde v kódu
   <Skin><BG>Button1{Width:100;Height:100;Text:Send;Enabled:false;}</BG></Skin>
   ```

4. Reverzibilita - Funkce deleteAllSkinObjectUpdates vrací všechny změny objektu do stádia před připojením k systému Radical Chat (tedy do období, kdy aplikace pracovala pouze ve statickém kontextu). Pokud byl objekt vytvořen dynamicky a ve statickém kontextu vůbec neexistoval, pak je vymazán a uvolněn z paměti, jinak se jeho nastavení vrátí do stavu statického kontextu.

5.3 Sekvenční identifikátory

Sekvenční identifikátory jsou základním mechanizmem nezbytným pro korektní rozpletení sekvence příkazů pocházejících z nejvyšší vrstvy (Radical Web Service). Rezervaci sekvenčních ID (SEQID) je možné provést několika způsoby. Starší způsob používaný v první verzi frameworku přenechával rezervaci identifikátorů na stream serveru (Radical Bridge). Každá událost, která byla předána ke zpracování PHP, nebo ASP skriptu měla vyhrazeno okno 16 SEQIDů pro úpravu skinu. Tento způsob se ukázal jako nedostačující. Na jednu stranu příliš plýtval identifikátory (které mnohdy nebyly využity), na druhou stranu neumožňoval v těle libovolné události použít příkazy dynamického skinování na více než na 16 objektech.
Aktuální verze frameworku používá modernější způsob, který sází na Radical Chat SEQID síťovou službu. Jedná se o jednoduchý (synchronizovaný) skript, jež poskytuje bloky sekvenční identifikátorů pouze na požádání a v rozsahu, který vývojář potřebuje. Toto řešení je vhodné pro vysoce distribuované systémy. Neplýtvá SEQIDy a zároveň umožňuje rezervaci bloků libovolné velikosti.



Obrázek 19 - SEQID síťová služba

Jedinou nevýhodou současného řešení je prozatímní nemožnost realokace velikosti bloku. To bývá občas nutné - obzvláště v případě, kdy není možné předem určit potřebné množství nezbytných identifikátorů. Tento nedostatek by měla vyřešit příští verze lehce pozměněné služby Radical Chat SEQID provider, která před původní sekvenční ID přidává časovou značku události (2 bajty) a rozšířený identifikátor bloku (1 bajt).

5.4 Zpracování příkazů

Příkazy jsou v koncových klientských aplikacích (Radical Flash Chat) zpracovávány paralelně a v mnoha případech také nehierarchicky. Sekvenční identifikátory spolu se čtyřmi závaznými pravidly (viz. kapitola 5.2) poskytují dostatečně silný mechanizmus pro to, aby bylo možné sekvenci příkazů kompletně rozplést, některé příkazy ignorovat, jiné úmyslně přeskočit.

Níže je uveden názorný příklad:

Demonstrace sleduje dvě takřka paralelně spuštěné události.
Uživatel2 je členem skupiny1 a skupiny2.

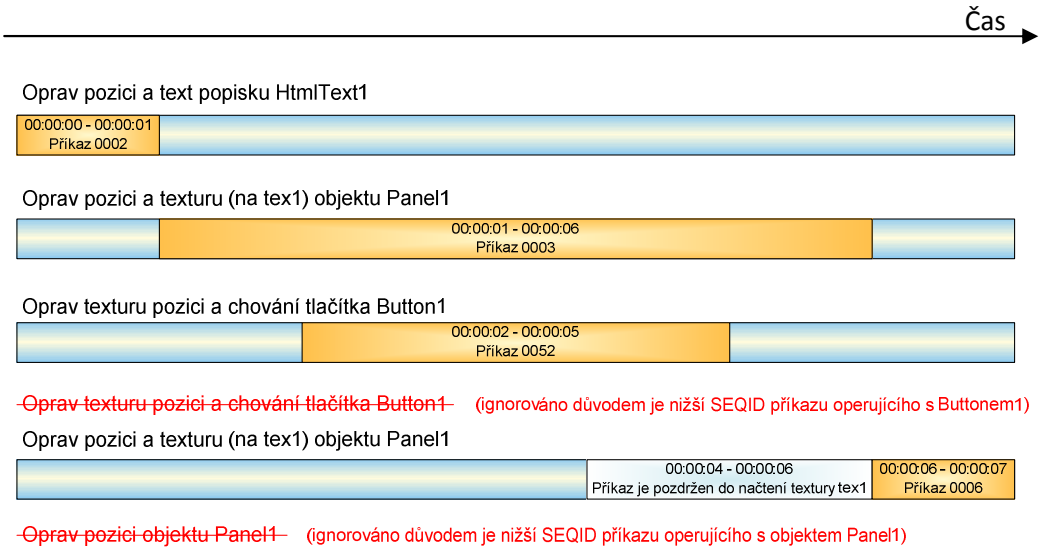
```
Událost 1
0001: Oprav pozici objektu Panel1 (pro skupinu uživatelů skupina1).
0002: Oprav pozici a text popisku HtmlText1 (pro skupinu uživatelů skupina2).
0003: Oprav pozici a texturu objektu Panel1 (pro skupinu uživatelů skupina2).
0004: Oprav texturu a chování tlačítka Button1 (pro uživatele3).
0005: Oprav texturu pozici a chování tlačítka Button1 (pro uživatele2).
0006: Oprav pozici a texturu bjektu Panel1 pro uživatele2.

Událost 2
0051: Oprav texturu a chování tlačítka Button1 (pro uživatele3).
0052: Oprav texturu pozici a chování tlačítka Button1 (pro uživatele2).
```

Pokud je stream server a Radical Bridge silně vytížen mohou příkazy dorazit do uživatelské aplikace v prohlížeči klienta v následujícím pořadí.

Příkazy, jak je přijal uživatel2:

```
0002: Oprav pozici a text popisku HtmlText1 (pro skupinu uživatelů skupina2).
        Časová prodleva 25 milisekund.
0003: Oprav pozici a texturu objektu Panel1 (pro skupinu uživatelů skupina2).
        Časová prodleva 50 milisekund.
0052: Oprav texturu pozici a chování tlačítka Button1 (pro uživatele2).
        Časová prodleva 150 milisekund.
0005: Oprav texturu pozici a chování tlačítka Button1 (pro uživatele2).
0006: Oprav pozici a texturu objektu Panel1 pro uživatele2.
        Časová prodleva 50 milisekund.
0001: Oprav pozici objektu Panel1 (pro skupinu uživatelů skupina1).
```



Obrázek 20 - zpracování příkazů dynamického skinování

Jak je vidět, předčasné a nehierarchické zpracování příkazů dává v konečném důsledku mnohem lepší výsledek a menší časové zpoždění, než kdyby se čekalo na kompletní synchronizaci celého příkazového bloku. Vnitřní logika je poměrně jednoduchá. Dorazí-li do uživatelské aplikace příkaz upravující chování nějakého objektu, který má nižší sekvenční ID než je aktuální revize daného objektu, pak je příkaz ignorován. Dorazí-li do uživatelské aplikace příkaz manipulující s objektem, který je právě opravován jiným příkazem s nižším sekvenčním ID - pak se vykonávání předchozího příkazu ihned zastaví a pokračuje se novým. To ale nemusí být vždy pravda. V rámci Radical Flash Chatu funguje jednoduchý rozhodovací algoritmus, který v některých případech pozdrží nově příchozí rozkazy do doby, dokud nejsou vykonány ty staré. Tento postup platí především pro objekty typu BasicPanel a BasicButton. U BasciPanelu je možné v rámci jednoho updatu opravit až 9 textur najednou. Pokud se tak stane a v krátké době poté dorazí další příkaz, který opravuje pouze 3 textury z 9 a zbylých 6 se shoduje s aktuálně nahrávanými, pak by bylo značně nelogické stornovat celý předchozí update. Stačí vyrušit nahrávání oněch 3 nerelevantních textur a pozdržet zpracování nově příchozího příkazu. Díky tomu se sníží objem přenesených dat a zvýší efektivita.

Na závěr je ještě nutné zmínit, jak funguje synchronizace skinu a chování místností u nově příchozích uživatelů. Jakmile se u klienta načte statický kontext aplikace a ta se úspěšně připojí ke stream serveru - vyžádá si Radical Flash Chat synchronizaci dynamického skinu. To spočívá v zapojení dvou shared objectů - vytvoření dvou vnitřních spojení, kterými Radical Brdige zasílá Radical Flash Chatu příkazy (private SO - privátní příkazy), (public SO - veřejné příkazy týkající se skupin nebo místností. Viz. obrázek 18). První věc, kterou Radical Bridge po připojení klientské aplikaci zašle je doplňkový XML dokument popisující rozšíření statického kontextu o změny (dynamického skinování) provedené v místnosti (či skupině) ještě před tím, než se klient do

systému připojil. Zaslány jsou pouze nejaktuálnější změny objektů, změny s největším revizním číslem.

5.5 Chyba upde-deup

Chyba “update before delete - delete before update” zkráceně upde-deup se objevila v první veřejně uvolněné verzi Radical Chatu. Vyskytovala se naprosto nepravidelně, zřídka, na první pohled nelogicky, nebylo ji možné tracovat (stopovat) a ve výsledku působila fatální komplikace a značnou nekonzistenci systému. Její odhalení trvalo téměř 3 měsíce. Zpočátku bylo známo, že se problém nachází v kódu dynamického skinování, avšak nevědělo se ve které části (Radical Flash Chat, Bridge, Web Service) a čeho se konkrétně týká. Chyba byla nakonec nalezena ve špatné analýze a návrhu funkce deleteAllSkinObjectUpdates (který jak známo vrací objekt a jeho funkcionalitu zpět do původního stavu načteného ze statického kontextu - včetně identifikátoru SEQID).

Níže je uveden jednoduchý příklad:

```
Vytvoř objekt1 pro skupinu uživatelů group1.
Uprav, změň barvu objektu1 pro skupinu uživatelů group1.
Smaž všechny změny provedené na objektu1 u uživatele2.
```

Příkazy mohou za určitých okolností dorazit k uživateli 2 v následujícím pořadí.

```
Vytvoř objekt1 pro skupinu uživatelů group1.
Smaž všechny změny provedené na objektu1 u uživatele2.
Uprav, změň barvu objektu1 pro skupinu uživatelů group1.
```

Jelikož byl (v první verzi) Radical Chat navržen špatně a snažil se všechny příkazy dynamického skinování včetně mazání zpracovávat ještě před tím než bylo známo jejich finální pořadí. Mohlo se za určitých vyjímečných okolností stát, že se objekt1 (z příkladu výše) zobrazil všem uživatelům stejně. Na této jednoduché chybě se pak kupily chyby další a nekonzistence nabývala na rozměru.

Aktuální verze Radical Chatu si již s problémem dokáže jednoduše poradit. Bylo zavedeno dynamické pole “skinObjectsArray_Deleted”, které udržuje informace o předčasně doručených příkazech typu deleteSkinObject. Radical Flash Chat pak s jejich vykonáním vyčká do doby, dokud skutečně nepřijdou na řadu.

6 Výkonnost, stabilita a spolehlivost

Obecně známá pravda říká, že skutečnou stabilitu a spolehlivost produktu prověří až čas. Proti tomuto tvrzení se nedá takřka nic namítnout, nicméně i přesto se Radical Chat (otestovaný stovkami uživatelů) dříve potýkal s drobnými problémy v oblasti spolehlivosti. K chybám docházelo a může docházet úpravami jádra systému Radical Bridge a zásahem do kódu Radical Web Service. Právě proto je kladen zvláště velký důraz na testování zpětné kompatibility některých vybraných funkcí.

6.1 Spolehlivost a stabilita Radical Bridge

V současné době je po každé větší úpravě systému provedeno na 118 funkčních testů, které ověří, zda přidáním nové funkcionality nevznikly nenadálé chyby v jiných částech frameworku. I když se 118 testů jeví jako velké číslo - nic nezaručuje, že se i přese všechno úsilí v testovacích verzích Radical Chatu nějaká ta chyba neobjeví.

Nejkritičtějším místem celého systému je Radical Bridge. Ve zdrojovém kódu je prakticky každá funkce uzavřena v bloku try-catch, nebo dokonce v několika do sebe vnořených bezpečnostních blocích. Díky tomu se podařilo zajistit, vysokou stabilitu Radical Bridge, který je schopen běžet celé měsíce bez zastavení a obsloužit desetitisíce klientů. Na druhou stranu, pokud je chování klienta nestandardní, nebo řídící aplikace Radical Web Service špatně implementovaná - může to skončit až odpojením klienta od systému. Stabilita Radical Bridge je tedy na opravdu vysoké úrovní, za cenu agresivního chování v případě selhání některé vnitřní funkce.

Další otázka, která se přímo nabízí zní: „jak je to se spolehlivostí ve spojení s hackery a jinými podvodnými živly?“ Skutečnost odhalila, že situace je nad očekávání dobrá. Volba unikátní třívrstvé architektury spolu s agresivním chováním Radical Bridge extrémním způsobem ztěžuje jakýkoliv efektivní útok na server. Radical Bridge působí podobně jako firewall. Důležité požadavky klientů filtruje a překládá na vnitřní události Radical Web Service. Odpověď od Radical Web Service putuje zpět do Radical Bridge, kde je nutné ji přeložit a teprve poté příkazy vykonat, přičemž množina příkazů je velmi malá. V praxi to znamená, že útočník nemůže přímo komunikovat s webovou službou (ať už je psána v PHP, nebo ASP.NET). Jediný systém, na který je možné po síti zaútočit je Radical Bridge. Ten jak známo běží pouze v operační paměti, není spojen s žádnou databází, neoperuje se soubory, ani dynamicky nenačítá žádné rozšiřující moduly. Pokud by se útočníkovi nějakým způsobem podařilo napadnout dobře zabezpečený Radical Bridge, pak by v nejhorším případě získal možnost odpojit ostatní klienty od sítě Radical Chat, nebo stáhnout

několik posledních vteřin jejich video-komunikace, která ještě zůstala uložena ve sdílené paměti RAM. V žádném případě ale nemůže operovat se soubory přímo na serveru, nebo nahrávat své vlastní skripty.

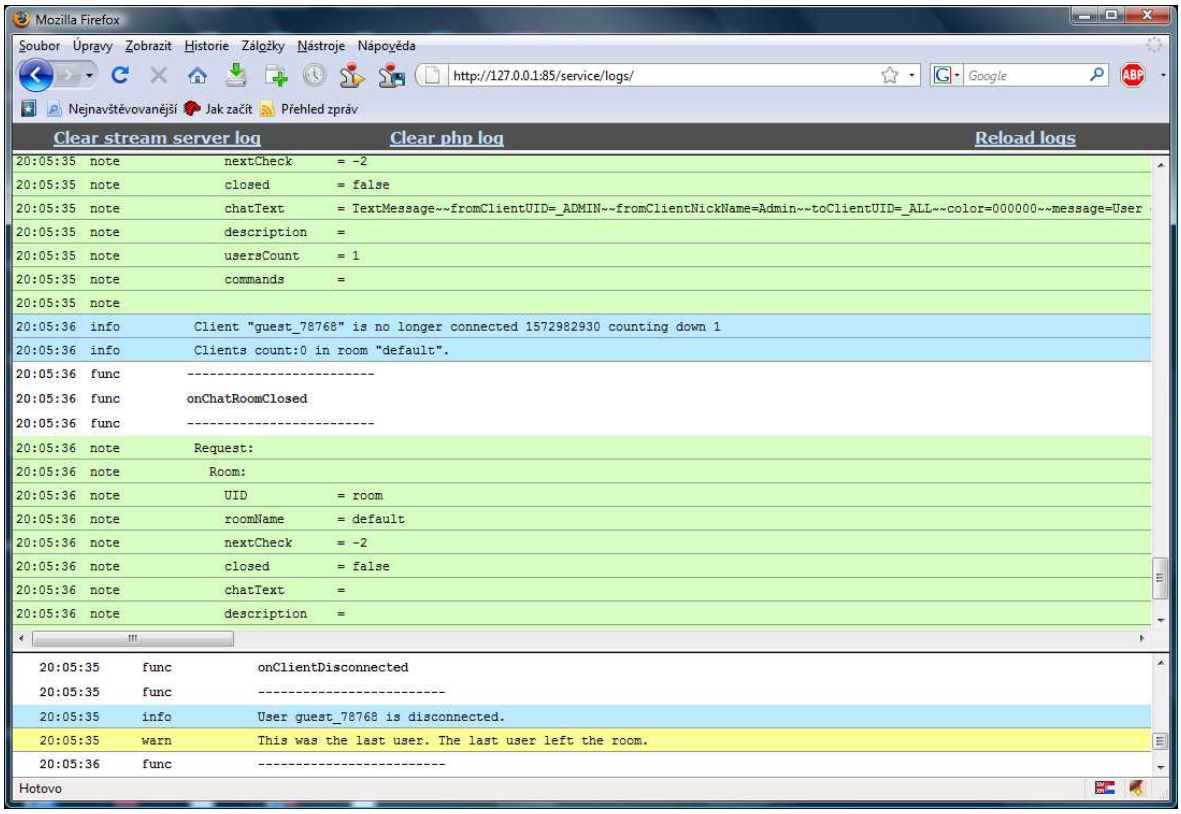
Jediný skutečný problém, který může ovlivnit spolehlivost Radical Bridge je zpětná kompatibilita aplikací psanými v dřívějších verzích Radical Chatu a případné změny v komponentách třetích stran jako je Adobe Flash Player, nová verze PHP6, atp.

6.2 Kompatibilita a interoperabilita

Radical Chat je vyvíjen postupně již od roku 2008. Skládá se ze 3 částí psaných v různých programovacích jazycích a v současné době má více než 30 000 řádků výkonného zdrojového kódu. Je velice těžké udržovat na takto rozsáhlém heterogenním systému plnou zpětnou kompatibilitu a interoperabilitu s ostatními subsystémy. Proto se autor frameworku rozhodl pro zajímavé (dvojí) řešení situace. V oblasti interoperability jsou podporovány celosvětově nejpoužívanější verze jednotlivých komponent (například aktuálně je to Adobe Flash Player 9, PHP5, již brzy C# .NET framework 3.0, …), tak aby uživatelé neměli nejmenší problém aplikaci spustit a vývojáři nebyli nuceni nasazovat na své servery neodzkoušené technologie. V oblasti zpětné kompatibility je to poněkud složitější. Praxe ukázala, že udržovat plnou zpětnou kompatibilitu na vysoce heterogenních systémech je velice obtížné, ba dokonce někdy i zcela nemožné. Na druhou stranu nutit vývojáře učit se s každou novou verzí nové API a kompletně přepisovat existující programy, by bylo krajně nerozumné. Proto tvurce frameworku zvolil zlatou střední cestu. Je li v jaderném API frameworku uvedena nějaká třída, funkce, nebo grafická komponenta, která provádí nějakou konkrétní činnost, pak je tato činnost zachována i ve všech následujících verzích frameworku. Název funkce se však může změnit, mohou přibýt nové parametry, třída může být rozdělena do několika logických celků a grafická komponenta nahrazena novějším typem. Všichni vývojáři (zákazníci) jsou včas informováni o případných změnách API. Mezi verzemi Radical Chat 0.8.5.1087 a Radical Chat 0.9.0.1324 existoval automatizovaný převodník statického kontextu aplikace (konfiguračního souboru config.xml), který sám dokázal převést starý konfigurační soubor na novější a nahradit zaniklé komponenty “Image” a “pattern” komponentou “BasicPanel”. Podobné automatizované převodníky budou pravděpodobně používány i v budoucnu. Pomohou vývojářům odhalit místa s funkcemi, které zanikly, nebo jejichž definice byla změněna a bude-li to možné, přímo navrhnou opravu. To zkrátí a výrazně zjednoduší upgrade finálních aplikací na nové verze MMVCF.

6.3 Logování a debugování

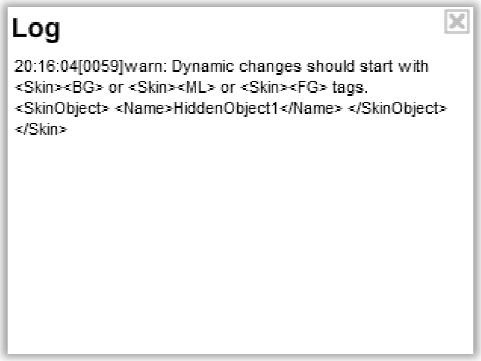
Většina vývojových prostředí obsahuje pokročilé nástroje na debugování kódu, které výrazným způsobem napomáhají odhalení chyb a zvýšení stability výsledných programů. Radical Chat je natolik komplikovaný, že debuger ve vrstvě Radical Web Service sám o sobě nic nevyřeší. Proto byl při tvorbě frameworku kladen důraz na logování jednotlivých událostí a veškeré komunikace mezi vrstvou Radical Bridge a Radical Web Service.



Obrázek 21 - stream server log

Díky tomuto typu logování je možné odhalit většinu chyb vyprodukovaných samotnými vývojáři ve vrstvě Radical Web Service. Bohužel zřídka se také může stát, že je chyba přímo v Radical Bridgi. Jelikož log soubor (jak je patrno z obrázku výše) obsahuje také informace o stavech virtuálních manipulátorů objektů před a po zpracování webovou službou - je eventuálně možné zpětně nasimulovat chování celého systému až do určitého kritického časového okamžiku a odhalit tak případné nedostatky v jádru frameworku.

Chyby mohou nastat i přímo v uživatelské části Radical Flash Chat. Většinou se jedná o syntaktické chyby XML kódu spojené s dynamickým skinováním. Tyto a podobné nedostatky je možné jednoduše odstranit za pomoci centrálního logovacího dialogu.



Obrázek 22 - Radical Flash Chat log

6.4 Chyby a nedostatky používaných komponent

Při vývoji systému Radical Chat bylo objeveno několik velice závažných chyb a nedostatků ve zdrojových kódech a komponentách třetích stran.

6.4.1 Chyba sdílení textových řetězců

Pravděpodobně nejzávažnější z nich je dvojitá chyba sdílení AMF3 textových řetězců mezi Wowza Media Serverem a Adobe Flash Playerem. AMF3 (Action Message Formát verze 3), který slouží výhradně k serializaci objektů a předávání informací mezi Adobe Flash Playerem a Flash Media Interactive Serverem byl nově implementován také do Wowza Media Serveru a Red5. Specifikaci AMF3 je možné nalézt na http://download.macromedia.com/pub/labs/amf/amf3_spec_121207.pdf. I když text na webových stránkách společnosti Wowza Media Systems uvádí, že podpora AMF3 byla plně zakomponována do nejnovějšího media serveru (verze 1.7), není tomu tak. Při vývoji univerzálního MMVCF autor této práce odhalil, závažnou chybu. Během sdílení dlouhých textových řetězců (například pomocí Shared Objektů), ve kterých se vyskytují speciální národní znaky (Čeština žčřš, nebo Finština äü) dojde mezi serverem WMS a Adobe Flash Playerem (verze 9.0.124 a 10.0.32) k nenadálé události – končící až pádem operačního systému na kterém je Adobe Flash Player spuštěn.

Pomocí AMF3 objektu je teoreticky možné zakódovat textový řetězec dlouhý až 2^{28} znaků (přibližně 256 MB čistého textu). Wowza Media Server se bohužel dopouští závažné chyby. Pokud chce vývojář pomocí WMS poslat do Adobe Flash Playeru textovou zprávu v Češtině o délce 2^{15} a více znaků – nepodaří se mu to. WMS bohužel špatně spočítá binární hlavičku textového řetězce,

který posílá sítí. Adobe Flash Player ihned po přijetí takto špatně zakódovaného textového řetězce zkolabuje. Jinak výborně napsaný a zabezpečený Adobe Flash Player není dostatečně dobře připraven na možnost nekorektní komunikace se stream servery třetích stran přes dříve úzkostlivě utajovaný protokol RTMP. Tato chyba byla několikrát testována na různých operačních systémech.

- Windows XP a Internet Explorer 6 – z 20 pokusů se operační systém dva krát zhroutil do modré obrazovky, ve zbylých 18 případech byl zaznamenán pád všech instancí webového prohlížeče.
- Windows Vista a Windows 7 – nejpoužívanější internetové prohlížeče Internet Explorer verze 8, Mozilla Firefox verze 3.0.14, Google Chrome verze 4.1, Safari verze 4.0.4, Opera verze 9 po testu výše zmíněné chyby Adobe Flash Playeru (verze 9.0.124 a 10.0.32) vždy zamrznou, nebo se zhroutí.
- Linux Gentoo 2009 a Firefox 3.0.14 – webový prohlížeč se vždy zhroutí.

Zjištěná závada byla nahlášena společnosti Wowza Media Systems a Adobe, ale doposud nebyla odstraněna.

Je s podivem, že i když se vývojáři Red5 a Wowza Media Serveru od sebe vzájemně distancují a na webových diskuzích vždy jedna i druhá strana svorně uvádí, že od konkurence neopisuje – tak se naprosto totožná chyba při kódování textových řetězců vyskytuje i v nejnovější verzi Red5 Media Serveru. Jak se říká v jednom Českém filmu „Stejné chyby, stejné …“.

6.4.2 Chyba snímání obrazu z více webových kamer

Další chybou Adobe Flash Playeru verze 9.0.124 a 10.0.32 je nekompatibilita provozu a přepínání mezi více webovými kamerami na jednom počítači. To zapříčinila pravděpodobně špatná spolupráce s lokalizovanými verzemi operačních systémů Windows Vista a Windows 7. Vše naprosto zřetelně demonstruje následující obrázek.

Obrázek 23 – výběr kamery

Pokud je v systému přítomna noname webová kamera jejíž ovladače ještě nebyly nainstalovány, zobrazí ji Adobe Flash Player pod zkomoleným názvem “Zobrazovací zařízení USB”. Má-li uživatel v systému dvě takové webové kamery nastává závažný problém. Po každém pokusu o výběr jedné

z nich generuje Adobe Flash Player interní chybovou hlášku a neprovede žádnou akci. Jelikož aplikační programové rozhraní nedovoluje výběr zaznamenávacího zařízení podle číselného indexu, nýbrž pouze podle jména, jsou vývojáři doslova odkázáni na milost a nemilost operačního systému a zařízení, které vybere jako výchozí. Přepnout se mezi kamerami zkrátka není možné.

I tato chyba byla nahlášena společnosti Adobe.

6.4.3 Neukončená spojení a Internet Explorer

Poslední chybou, která stojí za zmínku v této kapitole je závažný nedostatek internetového prohlížeče Internet Explorer verze 6. Tuto chybu je možné pozorovat v případě, že je otevřeno více oken prohlížeče a v jednom z nich spuštěn ActiveX prvek Adobe Flash Player. Jakákoliv aplikace, včetně Radical Chatu běží normálně až do okamžiku uzavření okna (například uživatel stiskne křížek, nebo klávesovou zkratku Alt+F4). Jakmile se tak stane není ActiveX prvek zrušen (uvolněn z paměti). Internet Explorer neuzavře RTMP spojení se stream serverem a ignoruje veškerou další příchozí komunikaci na portu 1395. Stream server si tak mylně myslí, že je klient stále připojen, což může narušit konzistenci aplikací.

Tento problém je všobecně znám již od roku 2003. Jelikož se jím společnost Microsoft v minulosti opakovaně odmítla zabývat je velice nepravděpodobné, že by tak učinila nyní. Jedno z možných funkčních řešení nastínili vývojáři Adobe koncem roku 2004 – ukončení RTMP komunikace je nutné vyvolat z flashové aplikace těsně před uzavřením okna webového prohlížeče. Svou roli zde sehrává kooperace flashe a javascriptového kódu, který detekuje uzavření okna.

Dalším možným řešením je pravidelné odesílání aktivního ping kódu sítí na stream server, který tak bude mít možnost detekovat případné vytimeoutování aplikace.

7 Závěr

7.1 Zhodnocení dosažených výsledků

Po dvou letech práce byl vytvořen pravděpodobně vůbec první univerzální multimediální framework zaměřený na tvorbu videokonferencí. Uplatnění nalezne především v komerční sféře v prostředí e-learningu, online helpdesků, ale také v mobilních komunikacích a případně pro monitorování budov a veřejných prostranství. Ačkoliv jsou v současné době oficiálně podporovány pouze dvě platformy (PHP a C# ASP.NET), které mohou s frameworkem pracovat přímo, fakticky může jeho služeb využít téměř každý vývojář znalý technologie webových služeb. Mezi nejvíce inovativní vlastnosti patří unikátní třívrstvá architektura používající technologii webových služeb zcela nestandardním způsobem, RPSF poziční systém a možnost dynamického skinování. Veřejná beta verze produktu Radical Chat 1.0.0.1455 byla uvolněna začátkem února 2010 a za dva měsíce zaznamenala 1500 stažení. Autor může s radostí oznámit, že projekt splnil všechna očekávání do něj vložená.

7.2 Novinky v experimentální fázi

Radical Chat se neustále vyvíjí. V červnu roku 2010 začal vývoj na nové experimentální části nejnižší vrstvy Radical Flash Chat, která využívá k přenášení živého audia a videa peer-to-peer spojení. Další novinkou je podpora zaznamenávání videa ve formátu VP6 a VP8. V červnu roku 2010 ještě stále není jisté, zda-li budoucí verze Adobe Flash Playeru bude obsahovat i encodér VP8 videa, nebo zda-li půjde pouze o běžný dekodér jako je tomu teď v případě formátu VP6. Ať již se věci vyvinou v dobrém, či špatném směru projekt Radical Chat se začal rozrůstat o novou komponentu Radical Web Browser založenou na jádru WebKit. Nový jednoduchý Web Browser umožní lepší propojení uživatelské části Radical Flash Chat s operačním systémem klienta a dovolí tak navíc streamovat živé video ve formátu VP6 (v budoucnu i VP8) a umožní sdílet dění na obrazovce.

7.3 Budoucnost projektu

Projekt je zatím licencován formou “free for non-commercial use” - zdarma pro nekomerční použití. Klientů, kteří jej touto cestou využívají, bude pravděpodobně jen pár desítek. Platící klienti jsou v současné době pouze 3 - Regionální televize Polar (Česká republika), firma R.A.D Pictures Tampere (Finsko) a společnost New Talent Group BV (Holandsko). S posledně jmenovanou společností jsou aktuálně vedeny rozhovory o možné budoucnosti projektu. Z rozhovorů a rýsující se finální dohody vyplynulo, že příští verze produktu již nebude dostupná volně ke stažení (nebo alespoň jedna její část - Radical Bridge). Stream server, který tvoří most mezi klientskou aplikací a serverovou vrstvou Radical Web Service bude nadále provozován pouze jako služba - pod správou společnosti New Talents Group BV a datový přenos pro větší zákazníky zpoplatněn. I když budoucí verze projektu přinese možnost vytvářet peer-to-peer spojení mezi více počítači - stále budou existovat sítě chráněné firewallem, nebo za nepropustným routerem, které vyžadují přímé napojení na stream server. Navíc Radical Bridge provozovaný jako veřejná služba (pod jednou střechou) nabídne klientům stabilní, rychlé a kvalitní spojení. Další podstatnou změnou (která vyplynula z dění posledních měsíců - květen, červen 2010) je nutnost zavedení bugzilly a veřejného fóra pro vývojáře. Příští verze frameworku kromě peer-to-peer spojení pro přenos audio a video streamů přinese i lehce pozměněné, efektivnější API a řadu nových unikátních funkcí.

Seznam použité literatury a zdrojů

[1] – Wikipedie - Multimedia Framework [online], přeformulováno, posl. revize 2009-1-12 <http://en.wikipedia.org/wiki/Multimedia_framework>

[2] – Wowza media server [online], přeloženo a přeformulováno, posl. revize 2009-1-12 <http://www.wowzamedia.com>

[3] – Red5 [online], přeloženo a přeformulováno, posl. revize 2010-3-3 <http://osflash.org/red5>

[4] – Radical Chat Documentation [online], posl. revize 2009-30-11 <http://mirror3.cze.cz/radicalchat/documentation.pdf>

Přílohy

Součástí této práce je DVD, které obsahuje:

- Text této diplomové práce ve formátu PDF a DOCX.
- Projekt Radical Flash Chat - veřejná distribuce (včetně čtyř demonstrativních aplikací používajících Radical Chat API)
- Framework Radical Chat - zdrojové kódy.
- Dokumentace frameworku v angličtině.